UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

INGENIERÍA TÉCNICA INDUSTRIAL: ELECTRÓNICA INDUSTRIAL

PROYECTO FIN DE CARRERA

CONTROL DE UN TANQUE DE PRESIÓN MEDIANTE EL MODULO PID DE UN AUTÓMATA PROGRAMABLE

AUTOR: ÁNGEL NÚÑEZ VALLE

TUTOR: RAMÓN BARBER CASTAÑO

DIRECTORA: CONCEPCIÓN ALICIA MONJE MIRACHET

27 de junio de 2011

Universidad
Carlos III de Madrid

Departamento de ingeniería de sistemas
y automática

PROYECTO FIN DE CARRERA

CONTROL DE UN TANQUE DE PRESIÓN MEDIANTE EL MODULO PID DE UN AUTÓMATA PROGRAMABLE

Autor: Ángel Núñez Valle

Tutor: Ramón Barber Castaño

Directora: Concepción Alicia Monje Mirachet

Leganés, junio de 2011

Agradecimientos

Sin lugar a dudas todo este trabajo no sería más que un mero proyecto si no fuera por muchas personas que me han dado fuerza y apoyo constante. Difícil se me hace poder resumir todo lo que han significado para mi estos años en la universidad, mucha gente querida, muchas historias, infinidad de momentos irrepetibles, anécdotas y sobre todas las cosas, la posibilidad de poder vivir y disfrutar de lo que hasta hoy es la actividad que me llena de ganas y alegría.

En primer me gustaría agradecer a mi familia que siempre me dieron todo cuanto han podido para que pudiese seguir adelante. Sin lugar a dudas a ellos les debo gran parte de lo que he logrado.

A lo largo de la carrera conocí a muchas personas maravillosas, pero entre todas ellas, me gustaría destacar a tres que sin ellas estos años no hubiesen sido lo mismo. A Juan y Rafa por esas magnificas películas después de los largos días de estudio en vuestras casas. También a la Cheno por ese agobio que transmite al estudiar.

A mi compañero, Salgado, por haber emprendido este proyecto juntos.

Y por último, a mis tutores Concha y Ramón por ofrecerme esta oportunidad y darme tan buenos consejos.

Ángel Núñez Valle

27 de junio de 2011

Índice general

Índice de figuras

Índice de tablas

1. INTRODUCCIÓN.

1.1. Motivación.

Como es sabido, el autómata nació como un paso adelante de los antiguos automatismos basados en relés. Incluso los primeros lenguajes de programación estaban basados en el simbolismo de contactos. No obstante, esta idea muy pronto se quedó corta al expresar todo el potencial de operaciones lógicas combinacionales y secuenciales y aritméticas que es capaz de ejecutar cualquier autómata. Así pues, el técnico que lo fuera en automatismos de relés debía aprender a utilizar los nuevos recursos que ponía a su alcance el autómata, recursos que fueron evolucionando muy rápidamente en los años 70 y principios de los 80, siguiendo la evolución de los microprocesadores y exigiendo esto su formación y puesta al día en una serie de aspectos, sobre todo de tipo electrónico o informático.

Los autómatas programables han supuesto la aplicación masiva del microprocesador al mundo de los controles industriales. Su gran ventaja ha sido que han permitido aplicar a dichos controles las conocidas ventajas de los sistemas programables. Pero quizás su mayor merito es que han permitido el uso generalizado del microprocesador por parte de no especialistas.

No obstante, el uso de autómatas obliga a adquirir nuevos conocimientos si se quiere obtener de ellos el máximo partido. Hay, sobre todo, dos aspectos a tener en cuenta. El primero es que el autómata obliga a pensar de forma distinta a la hora de plantear el diseño. El segundo es que el autómata permite disponer de comunicaciones con otros sistemas informáticos más potentes y esto amplía enormemente las prestaciones del conjunto.

Por otro lado se han aproximado al mundo del autómata una serie de técnicos muy formados en el mundo de la electrónica, la informática y los microprocesadores, pero con poca experiencia en el diseño de sistemas de control y un cierto desconocimiento de las condiciones de entorno en las que debe operar un sistema de control industrial. Para este tipo de técnicos el diseño basado en un ordenador de proceso resulta fácil hasta que llegan a la interfaz con el proceso o a la integración en el sistema de los captadores y accionamientos de potencia, para los cuales ciertamente los ordenadores normalmente no están preparados.

A pesar de lo anterior se debe aclarar que hoy en día no tiene sentido plantearse la disyuntiva «autómata u ordenador de proceso», sino que, tanto el autómata como el ordenador son piezas de un conjunto superior que los engloba y que se denomina CIM (Computer Integrated Manufacturing), donde se mezclan y se combinan los ordenadores, los controles numéricos, los robots y los propios autómatas, desempeñando cada uno ciertas funciones para las que está especialmente dotado [1].

Por ello, se plantea en este proyecto una arquitectura que aúna las técnicas de control con el mundo de la automatización, incorporando un PID industrial a un autómata programable que trabaje conjuntamente con un sistema SCADA.

1.2. Objetivos.

El presente proyecto fin de carrera es parte de un trabajo realizado conjuntamente con otro proyecto fin de carrera y cuyo objetivo global es:

- Diseñar un circuito neumático apropiado en el que se pueda controlar la presión del depósito.
- Desarrollar un sistema SCADA que controle la presión de un depósito neumático mediante el autómata S7300 de Siemens.

Como objetivos concretos para este proyecto se plantean:

- Diseñar e implementar el control de la presión de un depósito neumático, utilizando la función PID FB41 (CONT_C) que lleva integrada el software de programación STEP7 de SIEMENS.
- Realizar la programación y parametrización adecuada de la función FB41.
- Realizar una correcta conexión y parametrización de las E/S analógicas.

Primeramente se diseñará y ensamblará un circuito neumático con un acumulador neumático como protagonista y los elementos necesarios para los requerimientos de control. Se utilizará una electroválvula proporcional de 5 vías y 3 posiciones que permita pasar una cierta cantidad regulable de aire hacia el acumulador, mantener el aire, o por el contrario dejar escapar, en mayor o menor medida, el aire acumulado en el depósito. Para conocer en todo momento la presión acumulada precisaremos de un transductor de presión-voltaje. Se dispondrá además de un manómetro para conocer de manera analógica la presión alcanzada, éste será muy útil en las primeras etapas de desarrollo del proyecto.

En segundo lugar, y para este proyecto, el último paso, se configura y programa la función PID (FB 41 CONT_C) que lleva integrado el PLC SIMATIC S7-300, que será el elemento de control. En otras palabras, se debe aplicar al circuito neumático un regulador PID ya diseñado, a través del PLC, que debe parametrizarse de acuerdo con las necesidades, en este caso, del circuito neumático. Esta parametrización y configuración se realiza a través de un PC.

El PC también será necesario para desarrollar el tercer paso del proyecto global, el sistema SCADA (Supervisory Control And Data Acquisition). Este tercer paso pertenece al proyecto **REALIZACIÓN DE UN SISTEMA SCADA PARA EL CONTROL DE PRESIÓN DE UN DEPÓSITO NEUMÁTICO**. Se podría decir en pocas palabras que el presente proyecto se centra en el autómata programable y el proyecto antes mencionado, en el sistema SCADA.

Se planteo diseñar un sistema SCADA debido, básicamente, a que se pueden poseer conocimientos avanzados de controladores, pero sin tener idea alguna de programar el autómata de SIEMENS presente en el laboratorio. Con este sistema el usuario interactuará controlando y supervisando la presión del acumulador neumático. Se configurará un panel de

operador intuitivo y sencillo en el que se pueda comandar la presión que se desee tener en el acumulador y se muestre un gráfico de la presión en función del tiempo comparando la presión real que se obtiene con la demandada por el usuario.

A pesar de lo arduo que puede llegar a ser la configuración y programación del autómata programable, para que se pueda lograr un control de la presión del depósito, se aprovechará este proyecto para detallar el funcionamiento de la Función **FB 41 CONT_C** con el fin de poder observar el resultado de la configuración introducida en las respuestas del sistema, representadas en el panel del autómata, y de este modo, comprender de una manera práctica y sencilla el funcionamiento de la función PID.

1.3. Estructura del documento.

Este documento técnico estará compuesto por siete capítulos en los que se desarrolla el contenido del proyecto final de carrera. A través de estos siete capítulos se pretende dar al lector un conocimiento sobre la tecnología utilizada, una descripción de los elementos empleados, la explicación de todos los pasos seguidos para configurar y programar el PLC, una demostración de los resultados obtenidos al finalizar el trabajo y posibles ampliaciones a realizar en un futuro.

El **primer capítulo** es una introducción al trabajo que se va a desarrollar y pretende poner en situación al lector respecto a los objetivos del trabajo.

En el **segundo capítulo** se presenta un breve resumen de los sistemas de control industrial actuales y de cómo se llego a estos. En este capítulo se presentan una serie de definiciones y cuadros comparativos que permitirán al lector situarse en la temática de este texto.

En el **tercer capítulo** se enumera y se describen los diferentes elementos físicos que hacen posible la realización de este proyecto. Aquí se detallarán las características principales y datos técnicos, su utilidad y qué tipo de conexiones tienen los diferentes componentes.

El **capítulo cuarto** se centra en el autómata programable del proyecto. En éste se describe el funcionamiento del autómata y sus módulos. A diferencia de los capítulos 2 o 3, en los que también se habla del autómata programable, en este capítulo se hace de forma más específica.

El **capítulo quinto** contiene todos y cada uno de los pasos seguidos para una correcta configuración, programación y puesta en marcha del sistema de control de la presión del depósito neumático. Aquí se definirá más a fondo la función PID teórica y la función PID que tiene integrada el autómata. Se darán todo tipo de detalles con el propósito de que cualquier persona que quiera reconfigurar el autómata con fines similares pueda hacerlo fácilmente.

En el **capítulo sexto** se exponen los resultados obtenidos del sistema de control de la presión del acumulador. Se ofrece una visión gráfica de cómo responde el sistema ante la configuración y programación del autómata programable.

En el **capítulo séptimo** se presentan las diferentes conclusiones que se han obtenido una vez terminado este proyecto, así como trabajos futuros en los que se puede incluir nuevas herramientas en el sistema en general.

2. INTRODUCCIÓN A LA AUTOMATIZACIÓN.

2.1. Introducción.

El concepto de control [1] es extraordinariamente amplio, abarcando desde un simple interruptor que gobierna el encendido de una bombilla o el grifo que regula el paso de agua en una tubería, hasta el más complejo ordenador de proceso o el piloto automático de un avión.

Podríamos definir el control como la manipulación indirecta de las magnitudes de un sistema denominado **planta** a través de otro sistema llamado **sistema de control**. La **Figura 2.1** muestra esquemáticamente un diagrama de bloques con los dos elementos esenciales: sistema de control y planta.

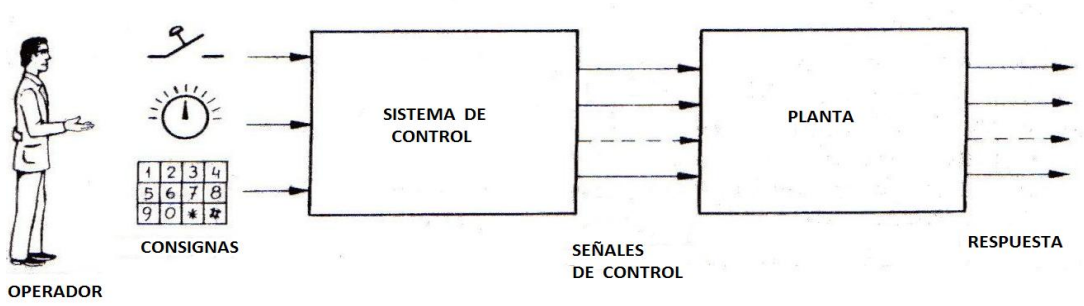

Figura 2.1. Sistema de control

Los primeros sistemas de control se desarrollaron con la revolución industrial de finales del siglo XIX y principios del siglo XX. Al principio, se basaron casi exclusivamente en componentes mecánicos y electromecánicos, básicamente engranajes, palancas, relés y pequeños motores, pero a partir de los años cincuenta empezaron a emplearse los semiconductores, que permitían el diseño de sistemas de menor tamaño y consumo, más rápidos y con menor desgaste.

En la década de los setenta, la complejidad y las prestaciones de los sistemas de control se incrementaron gracias al empleo de circuitos integrados y en particular los de tipo programable (sistemas basados en microprocesadores). Al tiempo que se desarrollaban los circuitos integrados lo hacían también los ordenadores digitales, si bien su empleo en la industria quedaba restringido al control de procesos muy complejos, debido a su elevado coste, necesidad de personal especializado para su instalación y manejo y a la poca facilidad de interconexión (interfaz) con el proceso, donde se manejan habitualmente tensiones y corrientes fuertes, para las cuales no suele estar preparado el ordenador.

La demanda en la industria de un sistema económico, robusto, flexible, fácilmente modificable y con mayor facilidad para tratar con tensiones y corrientes fuertes que la que

tenía el ordenador, hizo que se desarrollasen los autómatas programables industriales, abreviadamente *API* en la literatura castellana o *PLC* en la literatura anglosajona.

Los primeros autómatas pretendían, básicamente, sustituir a los sistemas convencionales con relés o circuitos lógicos, con las ventajas evidentes que suponía tener un hardware estándar. Por ello nacieron con prestaciones muy similares a las que ofrecían dichas tecnologías convencionales y sus lenguajes de programación eran muy próximos a los esquemáticos empleados en las mismas.

Estas limitaciones eran aconsejadas sólo por razones de mercado y no respondían a limitaciones tecnológicas de aquel momento, ya que las posibilidades que realmente podían ofrecer eran mucho mayores.

Los autómatas actuales han mejorado sus prestaciones respecto a los primeros en muchos aspectos, pero fundamentalmente a base de incorporar un juego de instrucciones más potente, mejorar la velocidad de respuesta y dotar al autómata de capacidad de comunicación. Los juegos de instrucciones incluyen actualmente, aparte de las operaciones lógicas con bits, temporizadores y contadores, otra serie de operaciones lógicas con palabras, operaciones aritméticas, tratamiento de señales analógicas, funciones de comunicación y una serie de funciones de control no disponibles en la tecnología clásica de relés. Todo ello ha potenciado su aplicación masiva al control industrial tal y como muestra la **Figura 2.2**.

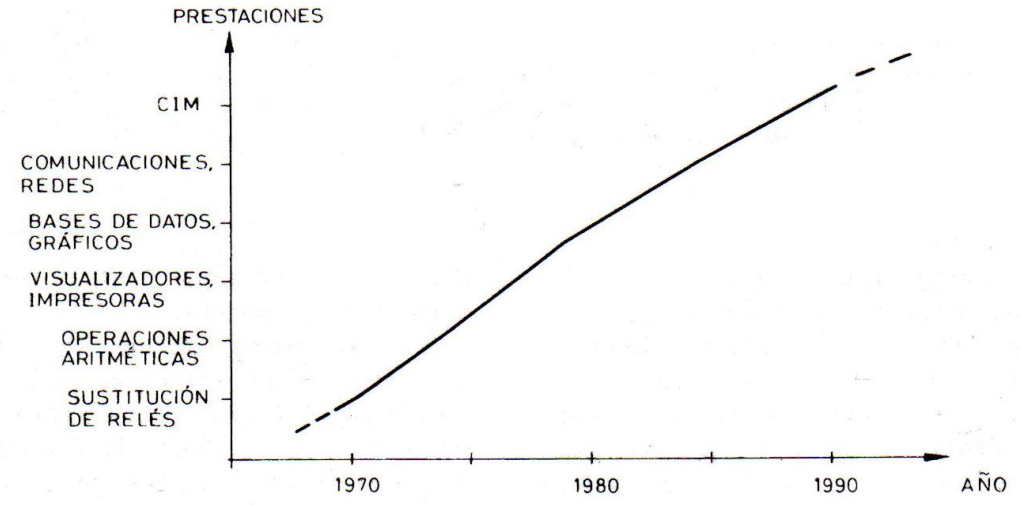

Figura 2.2. Evolución de las prestaciones de los autómatas

En definitiva, podríamos decir que los grandes autómatas actuales se acercan cada vez más a las prestaciones de un pequeño ordenador, siendo algunos incluso programables en lenguajes típicamente informáticos como el BASIC.

Sin embargo, la principal virtud del autómata sigue siendo su robustez y facilidad de interconexión al proceso y la tendencia actual no es precisamente la de acercarlo más a las prestaciones de los ordenadores en cuanto a su capacidad de cálculo, sino dotarlo de funciones específicas de control y de canales de comunicación para que puedan conectarse entre

sí y a los propios ordenadores. El resultado de esta integración es la red de autómatas conectada al ordenador, capaz de ofrecer las prestaciones y ventajas de ambos sistemas al integrar en un solo sistema todas las funciones de producción asistida por ordenador (CIM).

La disponibilidad de estos nuevos elementos y funciones en el campo del control industrial obliga a replantearse la configuración y los propios métodos de diseño de los automatismos.

2.2. Sistemas de control.

Según se ha indicado en la introducción, el objetivo de un sistema de control es el de gobernar la respuesta de una **planta**, sin que el operador intervenga directamente sobre sus elementos de salida. Dicho operador manipula únicamente las magnitudes denominadas de **consigna** y el sistema de control se encarga de gobernar dicha salida a través de los **accionamientos**.

El concepto lleva de alguna forma implícito que el sistema de control opera, en general, con magnitudes de baja potencia, llamadas genéricamente señales, y gobierna unos accionamientos que son los que realmente modulan la potencia entregada a la planta. Esta idea se refleja en la **Figura 2.3**.

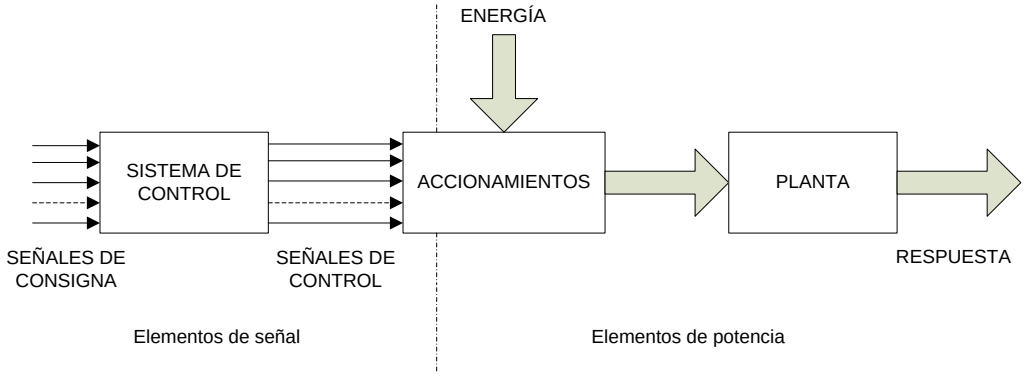

Figura 2.3. Sistema de control en lazo abierto.

Según la definición anterior, el conjunto de sistema de control y accionamientos se limitaría a ser un convertidor amplificador de potencia que ejecuta las órdenes dadas a través de las magnitudes de consigna. Este tipo de sistema de control se denomina **en lazo abierto**, por el hecho que no recibe ningún tipo de información del comportamiento de la planta.

Lo habitual, sin embargo, es que el sistema de control se encargue de la toma de ciertas decisiones ante determinados comportamientos de la planta, hablándose entonces de **sistemas automáticos de control.** Para ello se requiere la existencia de unos **sensores** que detecten el comportamiento de dicha planta y de unas **interfaces** para adaptar las señales de los sensores a las entradas del sistema de control. El diagrama de bloques será, en este caso, el de la **Figura 2.4**. Este tipo de sistemas se denominan **en lazo cerrado**, ya que su diagrama

muestra claramente una estructura con una **cadena directa** y un retorno o **realimentación,** formando un lazo de control.

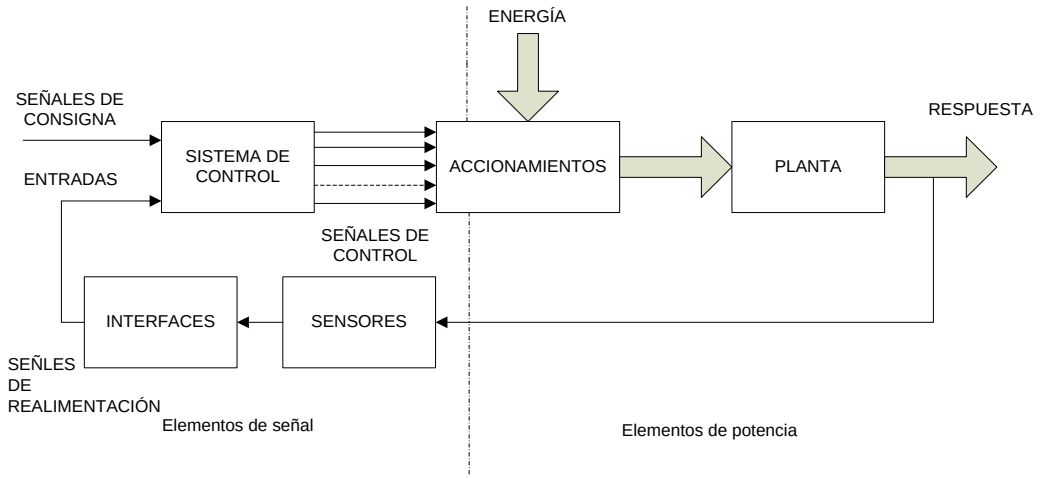

Figura 2.4. Sistema de control en lazo cerrado.

Así pues, en el caso más general, podremos dividir el sistema de control en los siguientes bloques:

- Unidad de control.
- Accionamientos.
- Sensores.
- Interfaces.

Cabe indicar aquí que el papel del autómata programable dentro del sistema de control es el de unidad de control, aunque suele incluir también, totalmente o en parte, las interfaces con las señales de proceso.

Al conjunto de señales de consigna y de realimentación que entran a la unidad de control se les denomina genéricamente **entradas** y al conjunto señales de control obtenidas **salidas.**

2.3. Automatismos analógicos y digitales.

Según la naturaleza de la señal que intervienen en el proceso, los sistemas de control pueden dividirse en los siguientes grupos:

- Sistemas analógicos.
- Sistemas digitales.
- Sistemas híbridos analógico-digitales.

Los sistemas analógicos trabajan con señales de tipo continuo, con un margen de variación determinado. Dichas señales suelen representar magnitudes físicas del proceso, tales

como presión, temperatura, velocidad, etc., mediante una tensión o corriente proporcionales a su valor (0 a 10 V, 4 a 20 mA, etc.).

Los sistemas digitales, en cambio, trabajan con señales todo o nada, llamadas también binarias, que sólo pueden presentar dos estados o niveles: abierto o cerrado, conduce o no conduce, mayor o menor, etc. Estos niveles o estados se suelen representar por variables lógicas o bits, cuyo valor puede ser sólo 1 o 0, empleando la notación binaria del álgebra de Boole.

Dentro de los sistemas digitales cabe distinguir dos grupos: los que trabajan con variables de un solo bit, denominados habitualmente **automatismos lógicos** y aquellos que procesan señales de varios bits, para representar, por ejemplo, valores numéricos de variables o contenido de temporizadores, contadores, etc. A estos últimos se les denomina genéricamente **automatismos digitales.**

Los sistemas de control actuales con un cierto grado de complejidad, y en particular los autómatas programables, son casi siempre híbridos, es decir, sistemas que procesan a la vez señales analógicas y digitales. No obstante, se tiende a que la unidad de control sea totalmente digital y basada en un microprocesador, que aporta la capacidad de cálculo necesaria para tratar las señales todo o nada en forma de bits y las señales analógicas numéricamente.

Dado que muchos de los sensores habitualmente empleados suministran señales de tipo analógico, las interfaces de estas señales deben realizar una conversión analógico-numérica, llamada habitualmente conversión analógico-digital (A/D), para que puedan ser tratadas por la unidad de control.

Puede ser necesario también disponer de señales analógicas de salida, para ciertos indicadores o para control de ciertos servosistemas externos. En tal caso el sistema de control debe disponer también de interfaces para la conversión digital-analógica (D/A), capaces de suministrar dichas señales a partir de los valores numéricos obtenidos por la unidad de control.

2.4. Componentes y modelos.

En los automatismos se encuentra habitualmente una diversidad de componentes ponentes o subsistemas de tipo mecánico, hidráulico, neumático, eléctrico o fisicoquímico. Se trata, pues, de sistemas que combinan múltiples tecnologías, haciendo necesario un lenguaje común para la coordinación e integración óptima de todas ellas en el sistema.

A nivel físico, la unión entre dichos subsistemas tecnológicamente diversos, la realizan los sensores e interfaces. Pero a nivel de caracterizar su comportamiento, el diseñador necesita un **modelo independiente de la tecnología** que le permita tratar a todos ellos con una metodología común, sea cual sea su principio tecnológico.

El modelo permite tratar a cada componente o subsistema como una «caja negra» a la cual se asocia una función de transferencia que relacione las magnitudes de salida de interés

con las magnitudes de entrada y que, por tanto, permita predecir su comportamiento una vez conocido su estado inicial y las señales de entrada aplicadas. Este enfoque nos permitirá, pues, tratar cualquier sistema o parte del mismo mediante un diagrama de bloques, que permite representar mediante un simbolismo común elementos de diversas tecnologías, que a pesar de sus diversas índoles aparecerán para el diseñador como homogéneo.

Para clarificar el concepto de modelo independiente de la tecnología se puede poner un ejemplo. Para el especialista en relés, el esquema eléctrico de un automatismo es un modelo a partir del cual es capaz de predecir el comportamiento del sistema ante determinadas entradas. Pero este modelo carece de significado para un especialista en hidráulica o neumática, que a su vez utiliza otro tipo de esquemas. Sin embargo, ambos tienen en común que emplean elementos todo o nada, que pueden representarse con el modelo común del álgebra de **Boole**, que sería el modelo independiente de la tecnología que permite tratar ambos tipos de sistemas bajo un mismo punto de vista.

De forma análoga, los sistemas analógicos pueden tratarse mediante funciones algebraicas continuas que relacionan las magnitudes de salida con las de entrada. Las herramientas matemáticas para el tratamiento de estos sistemas son básicamente la transformada de **Laplace**, para sistemas analógicos y la **transformada en z**, para sistemas digitales muestreados.

Los métodos del **álgebra de Boole**, la **transformada de Laplace** y la **transformada en z**, son útiles matemáticos imprescindibles para abordar el diseño de sistemas de control.

2.5. Automatismos cableados y programables.

Una de las claves del éxito de los autómatas programables frente a los equipos de relés, o incluso frente a equipos construidos a base de circuitos integrados, ha sido la posibilidad de realizar funciones muy diversas con un mismo equipo (hardware estándar) y cambiando únicamente un programa (software).

Atendiendo a este criterio podemos clasificar los sistemas de control en dos grandes grupos:

- Sistemas cableados (poco adaptables).
- Sistemas programables (muy adaptables).

Los primeros realizan una función de control fija, que depende de los componentes que lo forman y de la forma en que se han interconectado. Por tanto, la única forma de alterar la función de control es modificando sus componentes o la forma de interconectarlos.

Los sistemas programables, en cambio, pueden realizar distintas funciones de control sin alterar su configuración física, sino sólo cambiando el programa de control.

Las anteriores definiciones se deben matizarse algo más, puesto que, estrictamente hablando, cualquier equipo basado en un microprocesador es en principio programable, pero

para ello se requiere personal altamente especializado y equipos de desarrollo de cierta complejidad. En definitiva, del atributo «**programable**» se beneficia en este caso el fabricante del equipo, para el cual supone que con un hardware estándar puede variar dentro de ciertos límites la función del equipo; pero normalmente no está en la mano del usuario el poder alterar sus funciones, por lo que para este último el equipo es «**de programa fijo**» o «**adaptado a medida**».

En el autómata, el atributo «**programable**» hay que interpretarlo como «**programable por el usuario**», con lo cual éste obtiene los beneficios de un equipo multifunción con un hardware fijo. La base sigue siendo un equipo con un microprocesador, al cual se ha incorporado un programa **intérprete**, capaz de alterar la función de transferencia salida/entrada en razón de un **programa de usuario.** En realidad, se podría decir que esta es la característica más relevante que distingue al autómata programable de otros dispositivos o sistemas programables.

En la **Tabla 2.1** se ha resumido algunas ventajas e inconvenientes entre los tipos de automatismos.

CARACTERÍSTICA	SISTEMA CABLEADO	AUTÓMATA PROGRAMABLE
Flexibilidad de adaptación al proceso	Baja	Alta
Hardware estándar para distintas aplicaciones	No	Si
Posibilidades de ampliación	Bajas	Altas
Interconexiones y cableado exterior	Mucho	Poco
Tiempo de desarrollo del proyecto	Largo	Corto
Posibilidades de modificación	Difícil	Fácil
Mantenimiento	Difícil	Fácil
Herramientas para prueba	No	Sí
Stocks de mantenimiento	Medios	Bajos
Modificaciones sin parar el proceso («on line»)	No	Si
Coste para pequeñas series	Alto	Bajo
Estructuración en bloques independientes	Difícil	Fácil

Tabla 2.1. Comparación de los sistemas cableados y sistemas programables.

2.6. El autómata programable.

A lo largo de este capítulo se han ido clasificando los sistemas de control [1] según diferentes criterios, al tiempo que se iba situando a los autómatas programables dentro de cada una de estas clasificaciones. Ahora, pues, se procederá a dar una descripción de lo que se entiende por autómata programable.

Desde el punto de vista de su papel dentro del sistema de control, se ha dicho que el autómata programable es la unidad de control, incluyendo total o parcialmente las interfaces con las señales de proceso. Por otro lado, se trata de un sistema con un hardware estándar, con capacidad de conexión directa a las señales de campo (niveles de tensión y corriente industriales, transductores y periféricos electrónicos) y programable por el usuario.

Al conjunto de señales de consigna y de realimentación que entran en el autómata se les denomina genéricamente entradas y al conjunto de señales de control obtenidas salidas, pudiendo ser ambas analógicas o digitales.

El concepto de hardware estándar que venimos indicando para el autómata se complementa con el de modularidad, entendiendo como tal el hecho de que este hardware está fragmentado en partes interconectables que permiten configurar un sistema a la medida de las necesidades.

Así pues, encontramos autómatas compactos que incluyen una unidad de control y un mínimo de entradas y salidas y luego tienen previstas una serie de unidades de expansión que les permiten llegar hasta 128 o 256 entradas/salidas. Para aplicaciones más complejas se dispone de autómatas montados en rack con posibilidad hasta unas 2000 entradas/salidas controladas por una única unidad central (CPU). La **Tabla 2.2** resume a grandes rasgos las características de los autómatas actuales desde el punto de vista de modularidad.

AUTÓMATAS	COMPACTOS	MODULARES	
		CPU ÚNICA	VARIAS CPU
NÚMERO DE CPU	1 Central	1 Central	1 Central + *x* Dedicadas
Nº ENTRADAS/SALIDAS	8 a 256	128 a 1024	> 1024
JUEGO INSTRUCCIONES	Menor de 100	Menor de 100	> 100
PASOS DE PROGRAMA	Menor de 2000	Menor de 2000	2000 a 40.000
UNIDADES EXPANSIÓN	Digitales + Analógicas	Digitales + Analógicas	Digitales + Analógicas + Reguladores
FUNCIÓN EN RED	Esclavo	Esclavo	Maestro o Esclavo

Tabla 2.2. Características de los autómatas atendiendo a su modularidad.

Existe también la posibilidad, en autómatas grandes, de elección entre varios tipos de CPU, adaptados a la tarea que deba realizarse o incluso de múltiples CPU trabajando en paralelo en tareas distintas.

Así, las posibilidades de elección, tanto en capacidad de proceso como en número de entradas/salidas, son muy amplias y esto permite afirmar que se dispone siempre de un hardware estándar adaptado a cualquier necesidad.

Esta adaptabilidad ha progresado últimamente hacia el concepto de **inteligencia distribuida** (**Tabla 2.3**), gracias a las comunicaciones entre autómatas y a las redes autómata-ordenador. Esta técnica sustituye el gran autómata, con muchas entradas/salidas controladas por una única CPU, por varios autómatas, con un número menor de E/S, conectados en red y controlando cada punto o sección de una planta bajo el control de una CPU central. La muestra un resumen de las características comparadas de ambos sistemas.

CARACTERÍSTICA	AUTÓMATA ÚNICO	INTELIGENCIA DISTRIBUIDA
Capacidad de procesamiento	Buena	Óptima
Estructuración en bloques	Buena	Óptima
Facilidad de mantenimiento	Buena	Óptima
Almacenajes de mantenimiento	Altos	Menores
Disponibilidad del sistema frente a averías locales	Baja	Alta
Cableado	Grande	Reducido
Modularidad	Poca	Mucha
Coste de la instalación	Óptimo	Bueno
Posibilidades de modificación y ampliación	Buenas	Óptimas
Acceso a recursos compartidos	Rápido	Más lento
Rapidez de procesamiento	Buena	Óptima

Tabla 2.3. Comparación de sistemas con inteligencia distribuida frente al autómata único.

2.7. Sistema de control PID.

En la casi todos los sectores de la industria las variables a controlar son, muchas veces, las mismas: presión, caudal, nivel y temperatura. En el caso de este proyecto, la presión de un depósito neumático, será controlada por la función **PID** que está integrada en el software del autómata programable **SIMATIC S7-300**, el cual se detalla más en el capítulo 5.

El algoritmo de control más ampliamente extendido es el **PID** [2], pero existen muchos otros métodos que pueden dar un control de mayor calidad en ciertas situaciones donde el **PID** no responde a la perfección. El **PID** da buenos resultados en la inmensa mayoría de casos y tal vez es por esta razón que goza de tanta popularidad frente a otros reguladores teóricamente mejores.

Sea cual sea la tecnología de control, el error de regulación es la base a partir de la cual actúa el **PID** y se intuye que cuanto más precisa sea la medida, mejor se podrá controlar la variable en cuestión. Esta es la razón por la que el sensor es el elemento crítico del sistema. También se debe tener en cuenta la instalación, especialmente en la forma en que se transmiten los datos del sensor hacia el regulador y posibles fuentes de interferencias.

Un regulador **proporcional-integral-derivativo** o **PID** tiene en cuenta el error, la integral del error y la derivada del error. La acción de control se calcula multiplicando los tres valores por una constante y sumando los resultados. Los valores de las constantes, que reciben el nombre de constante proporcional, integral y derivativa, definen el comportamiento del regulador.

En el sistema de control en lazo cerrado mostrado en la **Figura 2.4** el sistema de control, teniendo en cuenta las tres acciones del **PID**, podría representarse de cómo en la **Figura 2.5**, en la cual vemos el diagrama de bloques correspondiente al **PID** y la representación matemática de la señal de control, que es la suma de las tres acciones, es decir, un valor proporcional al error **Kp**, mas la constante integral **Ki** por la integral del error, mas la constante derivativa **Kd** por la derivada del error.

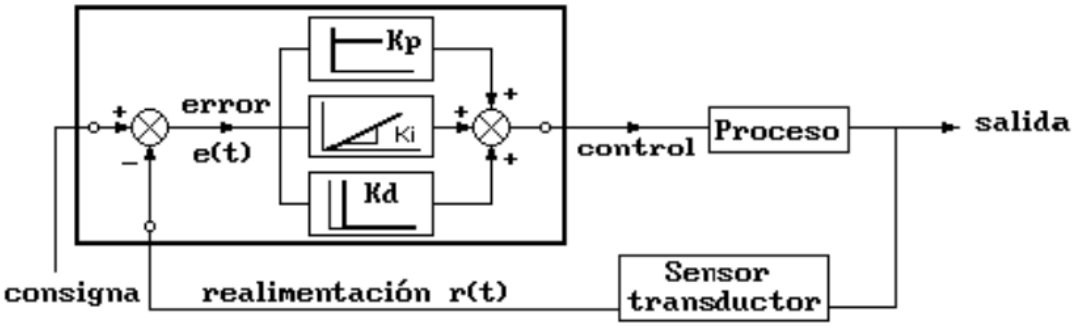

Figura 2.5. Acciones del regulador PID.

Esta acción combinada tiene la ventaja de cada una de las tres acciones individuales. La ecuación de un controlador con esta acción combinada se obtiene mediante:

$$\begin{pmatrix}\text{Señal de}\\ \text{control}\end{pmatrix} = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(t)dt + K_p T_d \frac{de(t)}{dt} \qquad (2.1)$$

$$e(t) = consigna - r(t) \qquad (2.2)$$

En donde **Ti** es el tiempo integral y **Td** el tiempo derivativo.

La **acción proporcional** hace que el **PID** [2] responda enérgicamente cuando el error es grande, condición que aparentemente es necesaria y suficiente, pero no es así en la mayoría de los casos por dos razones fundamentales:

- Muchas veces la variable regulada aumenta o disminuye si no existe una acción que la mantenga invariable, por ejemplo un cuerpo desciende por gravedad, un fluido disminuye

su nivel o presión si existe una vía de salida, un resorte tiende a adoptar la posición de mínima energía, etc. Cuando la variable se acerca al punto de consigna la acción proporcional se debilita y no vence la tendencia de la variable, alcanzando un reposo antes de lo previsto y por lo tanto manteniendo un error permanente.

- Aunque el error disminuye al aumentar la constante proporcional, no es correcto aumentar dicha acción todo lo necesario para conseguir un error muy pequeño, porque toda magnitud tiene cierta inercia a permanecer en su estado de reposo o de variación constante, es decir, que responderá desde el primer momento a la acción de control pero con cierto retraso, por ejemplo no podemos detener un móvil de forma instantánea, un motor no alcanza inmediatamente su velocidad nominal, etc. Si la acción proporcional es grande, la variable regulada se acercará al punto de consigna demasiado deprisa y será inevitable un sobrepasamiento.

Por la primera razón expuesta se deduce la conveniencia de añadir otra acción que responda **si el error se mantiene a lo largo del tiempo**, algo parecido a una memoria histórica que tenga en cuenta la evolución del error. Así actúa la **acción integral**, que se encarga de mantener una respuesta cuando el error se anula, gracias al error que existió en el tiempo ya pasado. Esta respuesta mantenida contrarresta la tendencia natural de la variable, en otras palabras, mejoraremos las características del sistema en régimen permanente.

Por la segunda razón expuesta, se comprende la necesidad de añadir otra acción que contrarreste la inercia del proceso, **frenándolo cuando evoluciona demasiado rápido y acelerándolo en caso contrario**, algo parecido a una visión de futuro que se anticipa a lo que previsiblemente ocurrirá. Así actúa la **acción derivativa**, conocida también como anticipativa por ese motivo. Esta acción mejorará la respuesta para el régimen transitorio.

La parte problemática es la sintonización de los parámetros del **PID**, es decir, dar valores a las constantes que representan las intensidades con las que actúan las tres acciones. La solución a este problema no es trivial puesto que depende de cómo responde el proceso a los esfuerzos que realiza el regulador para corregir el error.

2.8. El PC como controlador industrial.

El objetivo principal de la automatización [3] industrial consiste en gobernar la actividad y la evolución de los procesos sin intervención continua del operador humano.

Para conseguir este objetivo se tendrá que disponer de sistemas automatizados de control de procesos industriales con un alto grado de complejidad y autonomía de funcionamiento, y funciones adicionales a las básicas de ejecución de tareas y monitorización del proceso. Aspectos como la toma (automatizada) de decisiones, la gestión de los menús de producción, la generación de históricos, gestión de alarmas, etc., así como lo referente al control de calidad y mantenimiento, quedan cubiertos en los niveles de control de producción y supervisión de planta del modelo jerárquico de automatización.

Las funciones asociadas a estos niveles necesitan como requisitos imprescindibles el conocimiento de la realidad de la planta y la capacidad de interactuación sobre ella.

Los sistemas de interfaz entre usuario y planta basados en paneles de control repletos de indicadores luminosos, instrumentos de medida y pulsadores e interruptores cableados de forma rígida y con elevados costes de instalación y mantenimiento (**Figura 2.6**), que cubrían tradicionalmente estas necesidades, están siendo sustituidos por sistemas digitales (**Figura 2.7**) que utilizan la informática industrial para implementar el panel sobre la pantalla de un ordenador. Con una supervisión inteligente, que permite al operario interactuar con el proceso de forma dinámica, apoyándose en factores como la capacidad de almacenamiento y proceso del ordenador y su facilidad de comunicación con los controladores de planta, el operador conoce inmediatamente cualquier variación significativa del proceso mientras observa su evolución a lo largo del tiempo y sus probables tendencias.

Figura 2.6. Interfaz tradicional.

Figura 2.7. Interfaz digital.

En un sistema típico, el control directo de planta es realizado entonces por los controladores autónomos digitales y/o autómatas programables, mientras que el ordenador,

conectado con ellos, realiza las funciones de diálogo con el operador, tratamiento de la información del proceso y control de producción. En esta estructura, el ordenador no actúa directamente sobre la planta, sino que se limita a la supervisión y control de los elementos de regulación locales instalados en ella, además de procesar y presentar la información. Eventualmente, podría también ejercer acciones directas de control (lectura de sensores, activación/ desactivación de actuadores) por medio de un hardware adicional conectado a sus buses internos, aunque no es ésta la opción más frecuente.

El ordenador u ordenadores se apoyan en la estructura de dispositivos locales, uniéndose a ellos mediante líneas de interconexión digital (buses de campo, redes locales) por donde recoge información sobre la evolución del proceso (adquisición de datos), y envía las órdenes o comandos para el gobierno del mismo (control de producción): arranque, parada, cambios de producción, etc.

Los programas necesarios, y en su caso el hardware adicional que necesiten, se denominan en general sistemas **SCADA** [3] («Supervisory Control And Data Acquisition»).

Estos paquetes están ya en disposición de ofrecer unas prestaciones impensables hace una década:

- Posibilidad de crear paneles de alarma, que exigen la presencia del operador para reconocer una parada o situación de alarma, con registro de incidencias.
- Generación de históricos de señal de planta, que pueden ser volcados para su proceso sobre una hoja de cálculo.
- Creación de informes avisos y documentación en general.
- Ejecución de programas, que modifican la ley de control, o incluso el programa total sobre el autómata, bajo ciertas condiciones.
- Posibilidad de programación numérica, que permite realizar cálculos aritméticos de elevada resolución sobre la CPU del ordenador, y no sobre la del autómata menos especializado, etc.

Con ellas, se pueden desarrollar a aplicaciones basadas en el PC, con captura de datos, análisis de señales, presentaciones en pantalla, envío de resultados a disco o impresora, control de actuadores, etc.

Los paquetes SCADA suelen estar formados por dos programas: Editor y Ejecutor («Run-Time»), Con el primero se generan las aplicaciones descritas, aprovechando los editores, macros, lenguajes y ayudas disponibles y con el segundo se compilan para obtener el fichero EXE de ejecución continua tras la puesta en tensión o el arranque.

Para terminar se podría definir un sistema SCADA como una aplicación software especialmente diseñado para funcionar sobre ordenadores de control de producción con acceso a la planta mediante comunicación digital con los reguladores locales básicos, e interfaz

con usuarios mediante interfaces gráficas de alto nivel: pantallas táctiles, ratones o sensores, lápices ópticos, etc.

El sistema permite comunicarse con los dispositivos de campo (controladores autónomos, autómatas programables, sistemas de dosificación, etc.) para controlar el proceso en forma automática desde la pantalla del ordenador, que es configurada por el usuario y puede ser modificada con facilidad. Además, provee de toda la información que se genera en el proceso productivo a diversos usuarios, tanto del mismo nivel como de otros supervisores dentro de la empresa: supervisión, control de calidad, ingeniería y mantenimiento, etc.

3. MONTAJE DEL SISTEMA

3.1. Introducción.

El sistema tiene dos partes claramente diferenciadas, que serían el panel neumático (formado por filtro-regulador, distribuidor, manómetros, válvula proporcional 5/3, acumulador de aire comprimido y sensor analógico de presión), y un sistema de control, compuesto por un PLC que será el encargado de controlar el sistema y el PC, a través del cual se realizará la programación y configuración de éste (**Figura 3.1**).

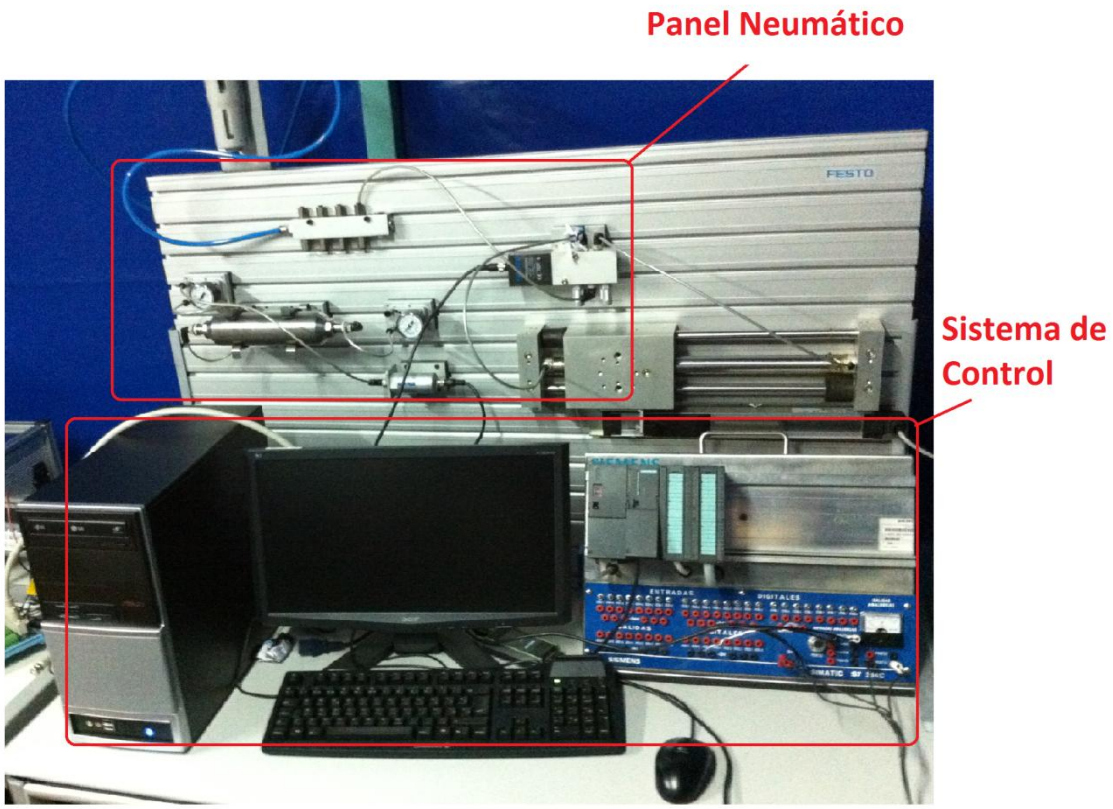

Figura 3.1. Sistema global.

3.2. Sistema neumático.

El sistema neumático va ensamblado tal y como se muestra en la **Figura 3.2** sobre una plataforma ranurada de aluminio mediante unos adaptadores que permite un fácil acoplamiento.

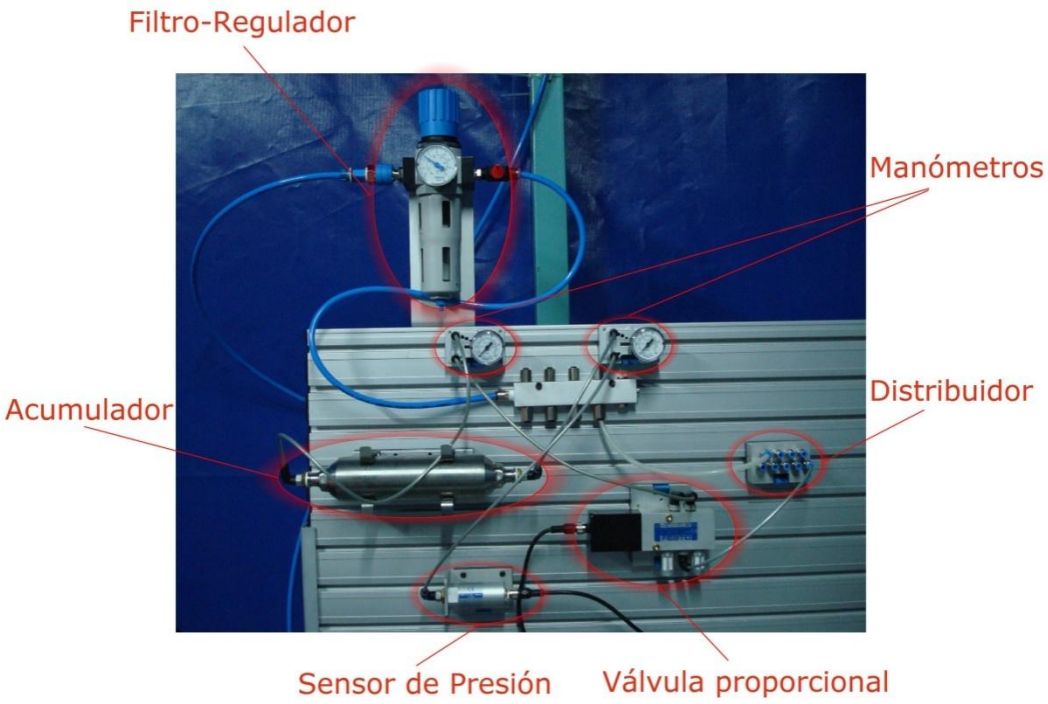

Figura 3.2. Montaje neumático.

Las interconexiones entre los dispositivos del montaje se pueden observar en el esquema de la **Figura 3.3**, dónde se puede apreciar que el depósito está controlado por una válvula de control proporcional, la cuál será la encargada de permitir el paso de aire al acumulador, o por el contrario de circularlo al escape. Los dos manómetros nos muestran visualmente la presión del depósito y el sensor envía en forma de señal eléctrica la presión que el aire ejerce sobre el acumulador neumático.

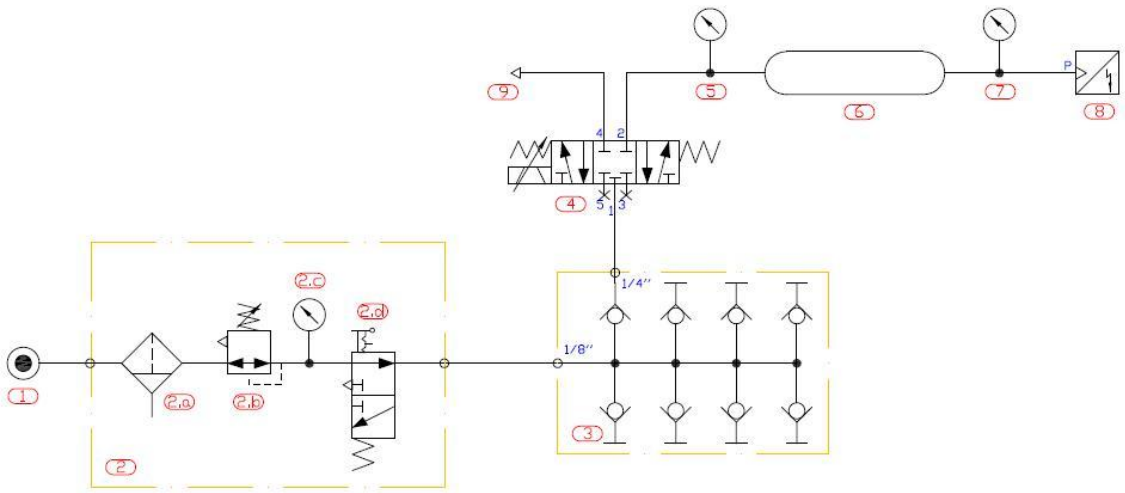

Figura 3.3. Esquema neumático.

A continuación se explicará cada elemento neumático, sus conexiones y sus principales características.

3.2.1. Filtro regulador.

Este elemento consiste en una unidad de filtración neumática y regulación de la presión, además dispone de una válvula de interrupción. En la **Figura 3.4** se muestra el aspecto físico del elemento.

Figura 3.4. Aspecto físico del filtro-regulador.

El sistema de filtrado [4] permite que el aire comprimido que entra al circuito no contenga partículas que puedan provocar un mal funcionamiento del equipo, limpiando cualquier tipo de suciedad, impureza, oxidación o condensación que pueda ser arrastrado desde el sistema de alimentación neumático. El regulador de presión ajusta el aire comprimido a la presión de funcionamiento del sistema y compensa cualquier fluctuación que se pudiese producir manteniendo una presión constante en todo momento. El cartucho de filtrado se puede drenar mediante un tornillo. El manómetro muestra la presión preestablecida. La válvula de 3/2 se acciona a través del mando azul deslizante. Este elemento queda representado en su conjunto por el esquema de la **Figura 3.5**.

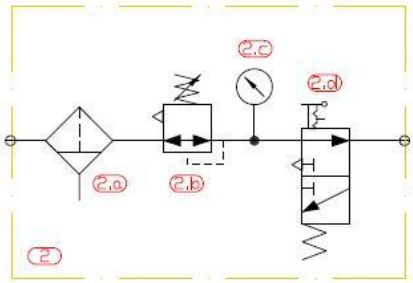

Figura 3.5. Esquema del filtro-regulador.

3.2.2. Distribuidor.

Como se puede ver en la **Figura 3.3**, El distribuidor adapta la conexión QS-6 que proviene del filtro regulador en ocho vías de trabajo tipo QS-4 con la que se interconectan el resto de elementos del sistema. Cada salida del distribuidor está protegida con un antirretorno. En la **Figura 3.6** se muestra una imagen del distribuidor.

Figura 3.6. Aspecto físico del distribuidor.

Como se puede ver en el símbolo hidráulico del elemento de la **Figura 3.7,** cada salida del distribuidor está protegida con un antirretorno, como elemento de seguridad, que evita que cualquier flujo de aire vuelva al circuito distribuidor.

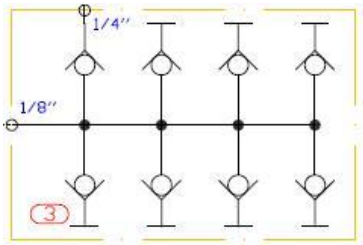

Figura 3.7. Esquema del Distribuidor.

Los tubos del sistema neumático serán en general PUN 4 x 0.75 con un diámetro exterior de 4 mm, interior de 2,6 mm con un espesor de 0,75 mm, exceptuando el que conecta el filtro-regulador con el distribuidor que será PUN 6 x 1 de 6 mm de diámetro exterior e interior de 4 mm con un espesor de 1 mm.

3.2.3. Válvula proporcional 3/5.

La servoválvula proporcional de 3 posiciones y 5 vías de accionamiento directo nos permite controlar el paso de una determinada cantidad de flujo de aire en una determinada dirección. El aspecto físico de la válvula proporcional del sistema es el mismo que el que se

muestra en la **Figura 3.8** a diferencia de que la válvula utilizada cuenta con racores rápidos de la serie QS en los terminales 1,2 y 4 para conexiones neumáticas G 1/8.

Figura 3.8. Aspecto físico de la válvula proporcional 5/3.

A través de una corredera regulada se obtienen tres posiciones distintas, la posición neutra o cerrada, la posición de llenado y la posición de vaciado o escape. La válvula también es capaz de modificar las secciones de las salidas para permitir el paso de una menor o mayor cantidad de aire. Las diferentes posiciones y secciones, y por tanto el control del flujo de aire tal y como se muestra en la **Figura 3** 9, se realiza electrónicamente mediante un transductor electromecánico a través de una señal de tensión de 0 a 10 V, correspondiendo a 5 V la posición neutra de cerrado, y a 0 y 10 V los valores máximos de escape o llenado respectivamente.

La válvula tiene una tensión de alimentación de 24 Vcc, una presión máxima de funcionamiento de 10 bar, una histéresis máxima de 0,4 % y un tiempo de conmutación de 5 ms. En caso de pérdida de la señal de control o sobrecalentamiento, la válvula quedará automáticamente en posición central de cerrado.

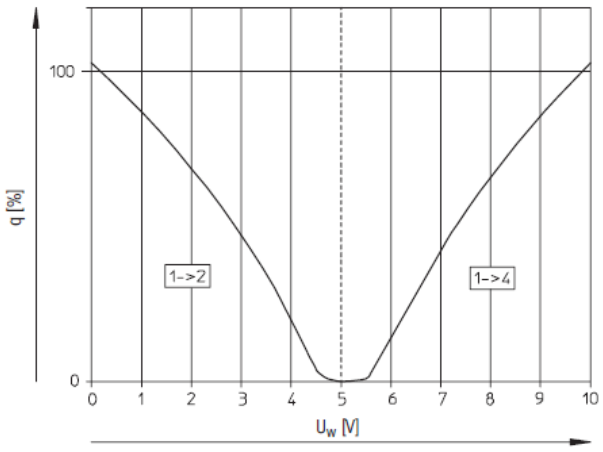

Figura 3 9. Caudal en función de la tensión.

Las conexiones eléctricas que permiten la alimentación y el pilotaje de la válvula se realizan mediante un cable de la serie KMPYE de Festo. El extremo conectado a la válvula es el mostrado en la **Figura 3.10**, el otro extremo cuenta con cuatro clavijas de conexión segura de diferentes colores que irán conectadas a la fuente de alimentación y a los pines analógicos del autómata programable. Estas conexiones se muestran en la **Tabla 3.1**.

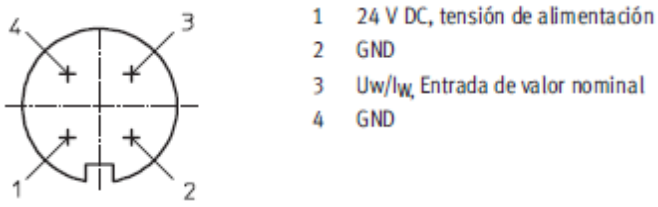

Figura 3.10. Conexiones eléctricas de la válvula.

Conexión	Terminal
Alimentación +24 V	Rojo
Alimentación 0 V	Azul
Señal de control (0 a 10 V)	Negro
GND control	Blanco

Tabla 3.1. Conexiones de la válvula proporcional 5/3.

En la **Figura 3.11** se muestra el esquema neumático de la servoválvula proporcional con pilotaje electrónico proporcional y retorno por muelle, dónde se pueden apreciar las 3 posiciones y las 5 vías, así como el pilotaje proporcional eléctrico con retorno por muelle.

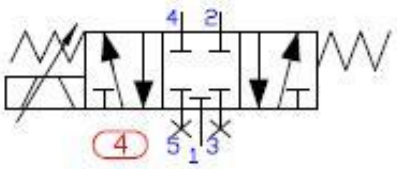

Figura 3.11. Esquema válvula proporcional 5/3.

3.2.4. Manómetros.

El sistema cuenta con dos manómetros a ambos lados del acumulador. Su aspecto físico es el que se muestra en la **Figura 3.12.** Su función, básicamente, es la de contrastar la información que se obtiene en tiempo real en el panel de operador del PC a partir de los datos transmitidos a la PLC desde el sensor de presión, y comprobar que estos son verídicos.

Figura 3.12. Aspecto físico del manómetro.

El manómetro puede medir presiones que estén en el rango de los 0 a 10 bar, pero no se recomienda superar para una presión continua el 75 % del valor máximo de escala. Su símbolo neumático es el mostrado en la **Figura 3.13.**

Figura 3.13. Esquema del manómetro.

3.2.5. Acumulador neumático.

El acumulador o depósito neumático mostrado en la **Figura 3.14**, es el dispositivo del sistema encargado de almacenar el aire. Este dispositivo de acero inoxidable puede acumular hasta 400 mililitros de aire y es capaz de soportar un rango de presión de -0,95 a 16bar.

Figura 3.14. Aspecto físico del acumulador neumático.

Generalmente estos acumuladores se utilizan con el fin de compensar oscilaciones de presión o como depósitos a los que recurrir para cubrir picos de consumo de aire, pero la función en este sistema es la de almacenar aire regular su presión. En la **Figura 3.15** se representa su simbología neumática.

Figura 3.15. Esquema del acumulador.

3.2.6. Sensor analógico de presión.

El elemento mostrado en la **Figura 3.16** está compuesto por el sensor de presión y un amplificador integrado. Va encapsulado con una carcasa de aluminio que lo aísla de la temperatura exterior. El flujo de aire que proviene de la salida del depósito, entra por una conexión neumática y ejerce presión sobre el sensor a través de una capa de silicona que protege al elemento piezorresistivo.

Figura 3.16. Aspecto físico del sensor de presión.

Se trata de un sensor que funciona bajo el principio físico del efecto piezorresistivo (el mismo principio que rige a las galgas extensiométricas). Este efecto se basa en un cambio de la resistencia cuando el material se ve sometido a un esfuerzo mecánico que lo deforma, en este caso compresión. El cambio de la resistividad se debe a la variación de la distancia interatómica del metal. Este tipo de sensores pueden verse afectados por la temperatura, por lo que hay que tener en cuenta el ambiente de trabajo donde se van a utilizar. En este caso no hay ningún problema ya que el sistema va a trabajar en todo momento a una temperatura ambiente dentro del rango aconsejado (0 a 85 ºC).

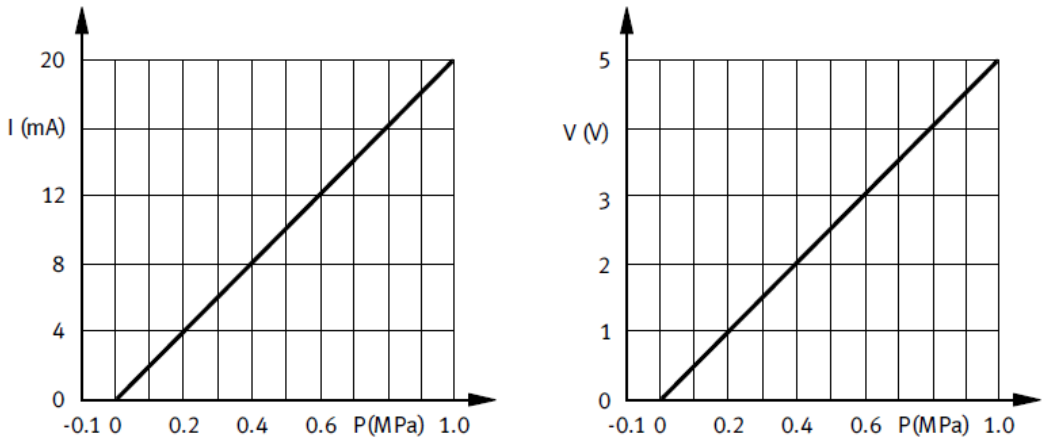

Figura 3.17. Corriente y tensión de salida en función de la presión.

La compresión producida sobre el sensor piezorresistivo provoca un cambio en la señal analógica que es incrementada por el amplificador obteniéndose una tensión o una corriente en función de esta presión, tal como se puede observar en la **Figura 3.17**, que va desde los 0 a los 10 bar (0 a 1 MPa). El sensor también es capaz de registrar presiones negativas, pero el fabricante no garantiza la linealidad para estos casos. La conexión eléctrica a través de la que se suministra corriente al sensor y por la cual se transmite la tensión y corriente de salida se hace a través de un terminal D.AS-SDE-K-4-GD de la casa Festo, mostrada en la **Figura 3.18**. Las conexiones para la fuente de tensión y las entradas analógicas de la PLC se realizan mediante cuatro clavijas de diferentes colores, indicadas en la **Tabla 3.** 2.

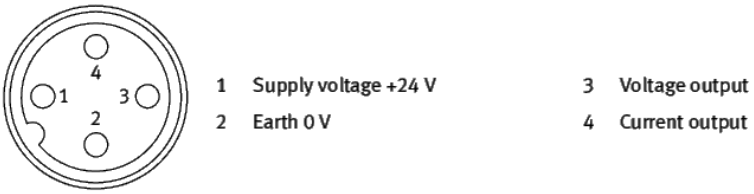

Figura 3.18. Conexiones eléctricas del sensor analógico de presión.

Conexión	Terminal
Alimentación +24 V	Rojo
Alimentación 0 V	Azul
Tensión de salida	Negro
Corriente de salida	Blanco

Tabla 3. 2. Conexiones del sensor analógico de presión.

El símbolo neumático utilizado para representar este tipo de sensores se representa en la **Figura 3.19.**

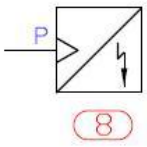

Figura 3.19. Esquema del sensor de presión analógico.

3.3. Controlador Lógico Programable (PLC).

Los sistemas de automatización de Siemens se componen de tres gamas que se distinguen por sus prestaciones. En primer lugar esta el SIMATIC S7-200 que es un Micro PLC compacto para la gama inferior. En segundo lugar el SIMATIC S7-300 que corresponde a la gama baja y media. En ultimo lugar esta la gama media alta y superior, el SIMATIC S7-400.

En este proyecto se ha utilizado el PLC SIMATIC S7-300 con CPU 314C-2 DP (nº ref. 6ES7314-6CG03-0AB0), mostrado en la **Figura 3.20.**

Figura 3.20. Aspecto físico de S7-300 314C-2 DP.

El siguiente capítulo se entrará en más detalles con respecto al autómata programable, su funcionamiento y sus módulos. En esta sección se enumeraran los siguientes datos técnicos, entre muchos otros, de esta CPU:

- 96 KB de memoria RAM (32000 instrucciones).
- Tiempo de ejecución:
  - 0,1 µs operaciones de bits.
  - 0,2 µs operaciones de palabras.
  - 2 µs operaciones de coma fija.
  - 3 µs operaciones de coma flotante.
- Contador S7 de 256 bytes, con rango de contaje de 0 a 999.
- Contador IEC tipo SFB.
- Temporizador S7 de 256 bytes, con margen de tiempo de 10 ms a 9,990 s.
- Temporizador IEC tipo SFB.

- 256 bytes de datos.
- Bloques: 1024 en total (DBs, FCs, FBs).
  - OB: capacidad de 16 Kbytes.
  - FB: máximo 512 (de FB0 a FB511).
  - FC: máximo 512 (de FC0 a FC511).
- Áreas de direccionamiento:
  - Digitales: 24 entradas (124.0 a 126.7), 16 salidas (124.0 a 125.7).
  - Analógicas: 4 + 1 entradas (752 a 761), 2 salidas (752 a 755).
- Lenguaje de programación: KOP/FUP/AWL.
- Funciones integradas:
  - Contadores: 4 canales, 24 V/60 kHz.
  - Frecuencímetro: 4 canales, máximo 60 kHz.
  - Salidas de impulso (PCM): 4 canales para modular ancho de pulso, máximo hasta 2,5 kHz.
  - Posicionamiento controlado: 1 canal.
  - Control SFB integrado: Regulador PID.
- Integra un puerto PROFIBUS DP maestro/esclavo.
- Alimentación:
  - 24 Vcc (margen admisible: 20,8 V a 28 V).
  - Consumo de corriente nominal: 800 mA.
  - Potencia disipada: tip. 14 W.

Para alimentar la CPU del S7-300 (además de al sensor de presión y la electroválvula) se precisa de una fuente de alimentación de 24 Vcc. Se ha utilizado la fuente de alimentación PS 307 2A (nº ref. 6ES7307-1BA00-0AA0) de la gama de productos que SIMATIC nos ofrece. Puede verse su aspecto físico en la **Figura 3.21.**

Figura 3.21. Aspecto físico de la fuente PS 307 2A.

Esta fuente de alimentación cuenta con las siguientes características eléctricas:

- Intensidad de salida 2 A.
- Tensión nominal de salida 24 Vcc., estabilizada, a prueba de cortocircuitos y marcha en vacío.
- Acometida monofásica (tensión nominal de entrada 120/230 Vca, 50/60 Hz).

- Separación eléctrica segura según NE 60 950.
- Puede utilizarse como fuente de alimentación de carga.

El PLC de este proyecto está instalado en un entrenador Maestro/Esclavo como el de la **Figura 3.22 .** En este entrenador las entradas y salidas van cableadas a distintos interruptores de simulación y hembrillas de 4mm con el objetivo de poner en práctica y simular procesos de una manera sencilla [5].

Figura 3.22.Entrenador Maestro / Esclavo de PROFIBUS.

3.4. El PC.

Para la realización del sistema de control de este proyecto ha sido necesario un ordenador con el que se podrá configurar y programar el PLC, mediante el paquete de software Simatic Step7. Una comunicación basada en un adaptador MPI a USB entre el ordenador y el PLC que a su vez trabaja como unidad remota junto a la instrumentación de campo formada por un sensor de presión y una válvula proporcional.

a) Unidad central.

La unidad central es básicamente un ordenador personal marca ASUS con las siguientes características:

- Tipo de procesador Intel® Pentium® Dual E2200 @2.20 GHz.
- Chipset Chipset Intel® G41 Express.
- Memoria RAM de 2 Gb.
- Unidades internas Disco duro SATA 3G de 640 GB (7200 rpm).
- Tarjeta gráfica ATI Radeon HD 4350 con tecnología Avivo.
- Puertos de E/S externos 6 puertos USB 2.0.
- Interfaz de red Interface de red Ethernet 10/100BT integrado. [8]

Este ordenador también servirá como soporte para la interfaz HMI, cuya representación sinóptica corre a cargo de una pantalla LCD de 19'' de marca ACER modelo X193HQ, mostrada en la **Figura 3.23.**

Figura 3.23. Pantalla LCD.

También estarán conectados un ratón y un teclado convencional a la CPU por puerto USB.

b) Comunicación.

La comunicación establecida entre el ordenador, MTU (Master Terminal Unit), y el autómata, RTU (Remote Terminal Unit) es de tipo topología punto a punto, consistente en una relación Maestro-Esclavo. Un solo elemento remoto, el PLC, está conectado al sistema de control, la CPU, mediante una línea de comunicación.

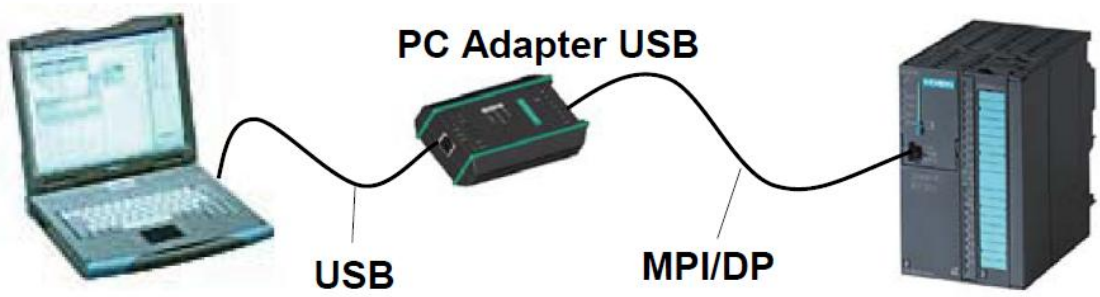

Figura 3.24. Comunicación CPU-PLC.

Esta línea de comunicación se basa en un adaptador que conecta el puerto USB del PC con el MPI/DP del S7-300, tal y como se puede ver en la **Figura 3.24**. Con este adaptador se evita tener que instalar alguna tarjeta adicional en el PC, que además es un problema en ordenadores no ampliables como los portátiles.

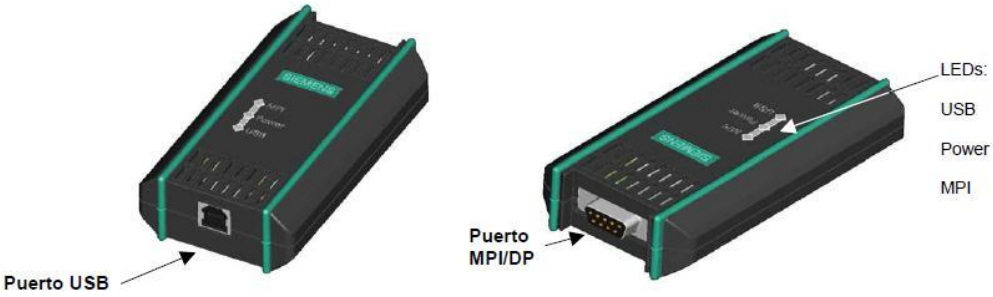

Figura 3.25. PC Adapter USB.

Este adaptador es el SIMATIC PC ADAPTER USB de la casa SIEMENS, mostrado en la **Figura 3.25.** Puede emplearse para la conexión de redes MPI, PROFIBUS y a partir de la versión 1.1 de firmware en redes PPI homogéneas, en la **Tabla 3.3** se muestran las velocidades de transferencia soportadas para cada tipo de red. Tiene las siguientes características:

- Detección automática del perfil de bus.
- Hasta 16 enlaces de comunicación, de los cuales, como máximo, 4 esclavo (enlaces DP/T).
- Soporte de routing (selección del camino por el que se envían los datos en una red de ordenadores).
- Soporta esclavos norma DPV1 (a partir de la versión firmware v1.3) mediante:
  - Asignación de direcciones de esclavo.
  - Diagnóstico de dispositivos.
  - Escribir/leer registro.

Velocidad de transferencia	MPI	PPI	PROFIBUS			
			DP	Estándar	Universal	Personalizado
9.600 bits/s	✗	✓	✓	✓	✓	✓
19.200 bits/s	✓	✓	✓	✓	✓	✓
45.450 bits/s	✗	✗	✓	✓	✗	✓
93.750 bits/s	✗	✗	✓	✓	✓	✓
187.500 bits/s	✓	✓	✓	✓	✓	✓
500 Kbits/s	✗	✗	✓	✓	✓	✓
1.500 Kbits/s	✓	✗	✓	✓	✓	✓

Tabla 3.3. Perfiles de bus soportados y velocidades de transferencia.

c) Unidades remotas RTU's.

Las RTU's son el conjunto de elementos dedicados a labores de control y/o supervisión del sistema, alejados del centro de control o MTU, y comunicados con éste mediante algún canal de comunicación. En esta clasificación se pueden distinguir varios elementos:

- RTU (Remote Terminal Unit): especializados en la comunicación.
- PLC (Programmable Logic Controller): tareas generales de control.
- IED (Intelligent Electronic Device): tareas específicas de control.

Las unidades RTU suelen ser ordenadores especiales que controlan directamente el proceso mediante tarjetas convertidoras adecuadas o que se comunican con elementos de control como PLC's o reguladores mediante los protocolos adecuados. De construcción más fuerte, operativos en un mayor rango de temperaturas que los ordenadores normales y mayor robustez eléctrica. Suelen manejar software elaborado en lenguajes de alto nivel como C o Visual BASIC, que les permite interpretar los comandos provenientes de la MTU.

Los PLC's con el tiempo han evolucionado convirtiéndose en módulos ampliables con distintas prestaciones que permiten configurar controladores apropiados para cada proceso. Entre ellos se encuentran los procesadores de comunicaciones que han hecho desaparecer a los RTU específicos, quedando estos incluidos en la CPU del PLC.

Los IED, llamados periféricos inteligentes capaces de tomar decisiones por sí mismos. En esta familia se encuentran elementos como los reguladores, variadores de frecuencia o transductores.

Siguiendo estas definiciones, los dispositivos de este proyecto que quedarían encuadrados en este tipo de hardware serían el PLC, y el sensor de presión.

4. ARQUITECTURA DEL AUTÓMATA PROGRAMABLE.

4.1. Introducción.

Los equipos Simatic S7 actualmente han superado ya la fase de maduración, por lo que se podria asegurar que es un sistema avanzado en el campo de la automatización de procesos mediante PLC.

Sin embargo, la rápida evolución del mismo tanto a nivel de software como de hardware en los últimos años, con sucesivas versiones que acercan cada vez más el entorno de programación de PLC's a los actuales entornos de programación de ordenadores (pese a que aún quedan grandes distancias que salvar, en especial en los procesos de depuración), han obligado a que no existiera aún un manual de referencia que permitiese adquirir en un periodo razonable de tiempo los conceptos al respecto de este nuevo sistema. Dichos conocimientos se encuentran repartidos en multitud de manuales de referencia, o simplemente no documentados.

En este capítulo se pretende dar una descripción detallada del PLC, de cada uno de sus bloques y de su funcionamiento.

4.2. ¿Qué es un PLC?

Un PLC (Programmer Logic Controller) o Autómata Programable Industrial es un equipo electrónico programable, diseñado para controlar en tiempo real y en ambiente industrial procesos secuenciales o combinacionales.

Actualmente han ensanchado su campo de aplicación introduciéndose en la automatización de viviendas y edificios (domótica). Los PLCs modernos tienen incorporados, además de las funciones de tratamiento lógico y secuencial, funciones de cálculo numérico, de regulación de procesos y de control de motores.

Una definición más formal es la siguiente: Un PLC es un sistema electrónico operado digitalmente, diseñado para trabajar en entornos industriales, que utiliza una memoria programable para almacenar instrucciones orientadas al usuario las cuales implementan funciones específicas de tipo lógico, de temporización, de contaje, aritméticas, y otras de carácter más específicamente relacionadas con tareas de automatización, con el objetivo de controlar, por medio de señales de entrada y salidas analógicas o digitales, varios tipos de máquinas o procesos.

Tanto el controlador programable como sus periféricos asociados están diseñados para poder ser fácilmente integrados dentro de un sistema de control industrial y para ser usados fácilmente.

Hasta no hace mucho tiempo el control de procesos industriales se venía haciendo de forma cableada por medio de contactores y relés. Al operario que se encontraba a cargo de este tipo de instalaciones se le exigía tener conocimientos técnicos para poder realizarlas y posteriormente mantenerlas. Además, cualquier variación en el proceso suponía modificar físicamente gran parte de las conexiones de los montajes, siendo necesario para ello un gran esfuerzo técnico y un considerable desembolso económico.

En la actualidad no se puede entender un proceso complejo desarrollado mediante técnicas cableadas. El ordenador y los autómatas programables han contribuido de forma considerable para que este tipo de instalaciones se hayan visto sustituidas por otras controladas mediante programa.

El PLC o Autómata Programable Industrial (API) nació como solución al control de circuitos complejos de automatización. Por lo tanto se puede decir que un API no es más que un equipo electrónico que sustituye los circuitos auxiliares o de mando de los sistemas automáticos. A él se conectan los sensores por una parte, y los actuadores por otra.

4.3. Evolución histórica

Los PLC´s [6] se introdujeron por primera vez en la industria en los años 60. La razón fue la necesidad que General Motors tenía de eliminar el gran coste en el que incurría al tener que reemplazar el complejo sistema de control basado en lógica cableada, relés y contactores cada vez que lo exigían las nuevas necesidades de producción de vehículos.

Los relés utilizados en aquella época eran dispositivos electromecánicos y poseían una vida limitada debido al desgaste de los contactos, a la vez que requerían un constante y estricto mantenimiento cuidadosamente planificado. Por otra parte, en grandes instalaciones industriales el número de estos dispositivos podía ser de cientos o miles y el cableado asociado tenía una longitud muy considerable, lo que implicaba un enorme esfuerzo de diseño y mantenimiento.

Los requerimientos de los nuevos controladores alternativos tenían que ser fácilmente programables por los ingenieros de planta o personal dedicado a su mantenimiento y caracterizarse por tener un tiempo de vida elevado. Además, los cambios en el programa tenían que realizarse de forma sencilla y como condición final, debían trabajar sin problemas en entornos industriales adversos (ruido electromagnético, polvo, ambientes corrosivos, falta de espacio, etc. La solución era el empleo de una técnica de programación que resultara familiar para los operarios y el reemplazamiento de los relés mecánicos por relés de estado sólido (tipo semiconductor).

Bedford Associates propuso como solución al problema de General Motors un dispositivo denominado Controlador Digital Modular (MODICON), mientras que otras compañías propusieron diferentes soluciones basadas en control mediante ordenador. La solución de Bedford fue la ganadora del concurso y el MODICON puede considerarse como el precursor de los PLCs actuales.

A mediados de los 70, los microprocesadores ya disponían de la potencia necesaria para resolver de forma eficaz la lógica de los pequeños PLCs existentes. Las necesidades de comunicación entre los PLCs comenzaron a aparecer en 1973. El primer sistema fue el bus MODBUS. El PLC necesitaba poder dialogar con otros PLCs y también necesitaba poder enviar y recibir señales de tensión variables entrando en el mundo analógico. Desafortunadamente, la falta de un estándar, acompañado de un continuo cambio tecnológico, hizo que la comunicación de PLCs fuera un maremágnum de sistemas físicos y protocolos propietarios incompatibles entre sí.

En los 80 se produjo un intento de estandarización de las comunicaciones con el protocolo MAP (Manufacturing Automation Protocol) impulsado por General Motors. También fue un tiempo en el que se redujeron las dimensiones del PLC y se pasó a programar con ordenadores personales en vez de hacerlo mediante los clásicos terminales de programación a pie de planta.

Los 90 han mostrado una gradual reducción en el número de nuevos protocolos establecidos y en la modernización de las capas físicas de los protocolos más populares que sobrevivieron a los 80. El último estándar (**IEC 1131-3**) intenta unificar el sistema de programación de todos los PLC en un único estándar internacional.

En la actualidad se busca la estandarización de los autómatas programables y sus periféricos, incluyendo los lenguajes de programación que se deben utilizar. Lo anterior se debe a que hoy en día aún siguen persistiendo sistemas de control específicos del fabricante, con programación dependiente y conexión compleja entre distintos sistemas de control. Esto tiene como consecuencias para el usuario costes elevados, escasa flexibilidad y falta de normalización en las soluciones al control industrial. Lo anterior se pretende eliminar con la norma, antes mencionada, **IEC 1131-3**, que busca hacer el trabajo independiente de cualquier compañía.

Otra tendencia internacional es la utilización de los lenguajes de alto nivel (Java, C++) en la programación de PLCs. Esta variante tiene el inconveniente de que requiere que el usuario tenga conocimientos sobre dichos superlenguajes, lo cual no es del dominio de la mayoría de los usuarios de PLCs. Lo que se pretende es mezclar la sencillez, que tanta popularidad han dado a estos equipos, de programar los PLC`s con las propiedades de estos lenguajes de alto nivel.

4.4. Principio de funcionamiento.

Los equipos programables [7] emplean un procesador binario que es capaz de interpretar una serie de códigos o instrucciones que especifican las acciones a realizar en función del estado de las variables del sistema. En la mayoría de los casos, el procesador puede interpretar una sola instrucción en cada instante, aunque lo hace a gran velocidad (µ segundos); esta forma de actuar introduce el concepto de tratamiento **secuencial** de la información, que se ilustra en la **Figura 4.1**.

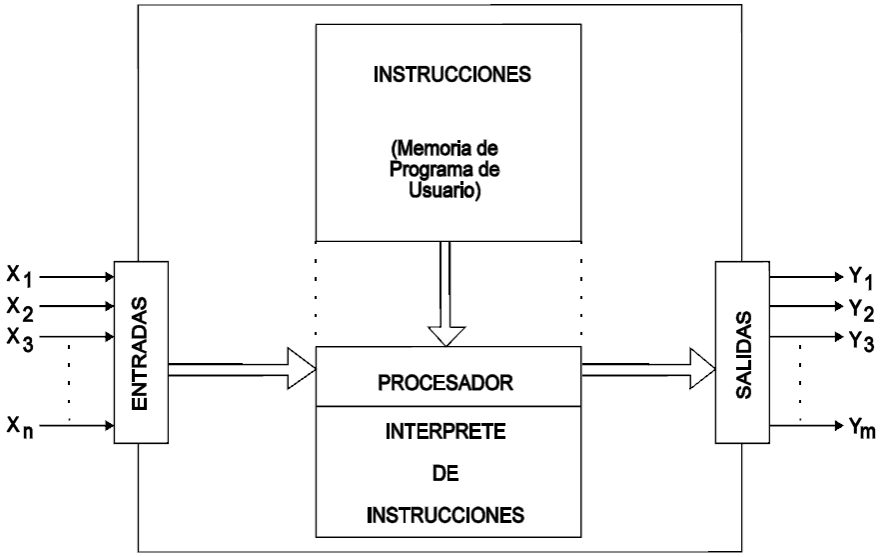

Figura 4.1. Tratamiento secuencial de la información en un sistema programable.

En el autómata las instrucciones se almacenan en una memoria, que recibe el nombre de **memoria de programa**. El procesador recoge los estados de las señales de entrada y los almacena en otra memoria denominada **tabla de E/S**, para su posterior empleo. Inicia entonces el acceso una tras otra a las instrucciones, que especifican un operando (variable) y la operación lógica a efectuar; en el curso de esta exploración de la **memoria de programa** se obtienen los resultados de las ecuaciones lógicas del sistema, y van siendo almacenados también en la **tabla de E/S**. Una vez finalizada la lectura del programa, tiene lugar la "actualización" de estados de E/S para lo que se transfieren a las salidas los resultados obtenidos de la comprobación de instrucciones, y se vuelven a almacenar los estados de las entradas. Tal como se indica en la **Figura 4.2**, este proceso se repite de forma indefinida mientras el equipo esté operativo. Dada la velocidad con que se realiza cada ciclo, se puede decir que las salidas se ejecutan en función de las variables de entrada prácticamente en **tiempo real**.

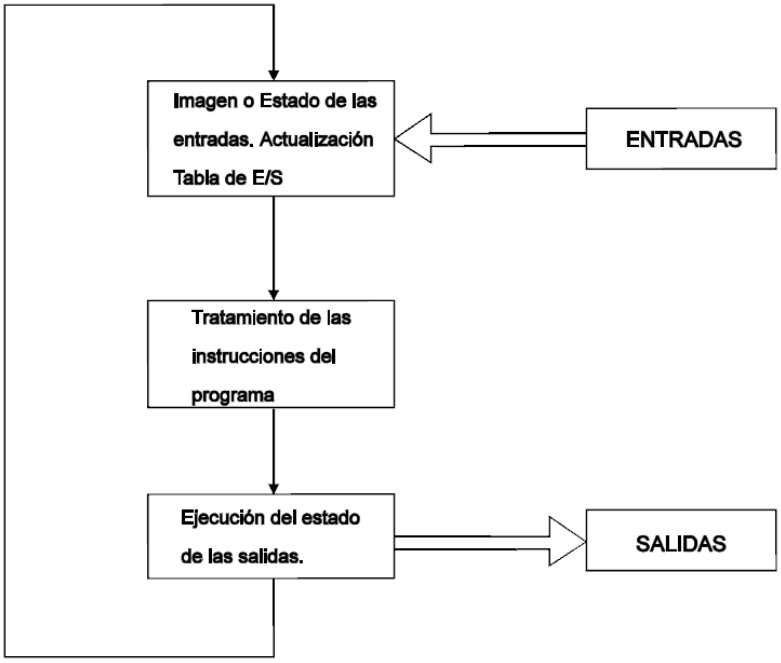

Figura 4.2. Ciclo básico de trabajo de un autómata.

Así pues, en un autómata programable existe un **tiempo de respuesta** (milisegundos, µ segundos), cuya magnitud es función de la cantidad y complejidad de las instrucciones que forman el programa y de la velocidad con que se ejecutan; durante la resolución del programa **el equipo ignora la evolución externa de la máquina o proceso**. En ciertas aplicaciones de evolución muy rápida esto puede llegar a ser un problema ya que llegará a perderse alguna información, y en consecuencia puede darse un funcionamiento anómalo. Este caso cada vez resulta más difícil, ya que los tiempos de duración de un ciclo básico de trabajo de un autómata suelen estar hoy en día a los valores que requieren las aplicaciones o procesos a controlar.

4.5. Estructura de un autómata programable.

Los equipos que responden al concepto de autómata programable Industrial, se presentan en diversas formas de construcción física y organización interna, pero todos están compuestos básicamente de los mismos elementos, que pueden dividirse en dos grupos:

- Estructura externa.
- Estructura interna

4.5.1. Estructura externa.

Se refiere al aspecto físico exterior [6] del mismo, es decir, los bloques o elementos en que está dividido. Todos los autómatas programables, poseen una de las siguientes estructuras constructivas:

- **Compacta**: Se usa para automatismos lógicos con equipos pequeños y donde las E/S (Entrada/Salidas) son fijas. En un solo bloque están todos los elementos, esto es, fuente de alimentación, CPU, memorias, E/S, etc.

- **Modular**: Se usa en PLC´s industrial. Posee módulos desmontables y es posible expandir los modulos de entradas y salidas. Existen dos configuraciones, la **estructura americana** que separa las E/S del resto del autómata y la **estructura europea** donde existe un modulo para cada función (fuente de alimentación, CPU, E/S). La unidad de programación se une mediante cable y conector. La sujeción de los mismos se hace bien sobre carril DIN o placa perforada, bien sobre RACK, en donde va alojado el BUS externo para unión de los distintos módulos que lo componen.

Respecto a su tamaño, como se puede observar en la **Tabla 4.1**, se suelen clasificar dependiendo de la capacidad de su memoria y del número de E/S de que dispongan.

Gama	Memoria de usuario	Nº de E/S
Baja	<4Kb	0 a 128
Media	4Kb a 16Kb	128 a 512
Alta	16Kb a 100Kb	>512

Tabla 4.1. Clasificación de los autómatas según su tamaño.

4.5.2. Estructura interna.

El Autómata Programable Industrial [7] es una máquina electrónica digital programable que está constituida por dos elementos básicos:

- La Unidad Central de Proceso (UCP o CPU).
- El sistema de Entradas y Salidas (E/S).

Con estos dos elementos, el equipo ya es operativo sobre la máquina o proceso a controlar, pero existen otros componentes que aunque no forman parte del autómata como equipo, son necesarios para su aplicación. Estos componentes, generalmente denominados **periféricos**, son los equipos de programación, las impresoras, los visualizadores, terminales, etc.

En los siguientes párrafos se hace una descripción de las características funcionales de los distintos componentes que pueden formar parte del autómata.

4.5.2.1 CPU.

La Unidad Central de Proceso de un Autómata comprende esencialmente: **el procesador**, la **memoria** y las **comunicaciones**, tanto con periféricos como con el sistema de **E/S**. Además, contará con sus **circuitos auxiliares asociados**. En algunos casos la CPU trae incorporada la **fuente de alimentación**, en el caso de que no sea así, será un modulo aparte. Obviamente, la forma constructiva varía de unos modelos a otros.

Desde el punto de vista funcional, la **CPU** es el corazón del autómata, realizando todas las tareas de control, tanto en lo que se refiere a adquisición de información y gobierno de los accionadores del proceso a controlar, como en lo que atañe a funciones internas de vigilancia del adecuado funcionamiento de los componentes del equipo.

Se puede observar en la **Figura 4.3** como dentro de la CPU tiene lugar un intercambio continuo de información entre los distintos componentes de la misma.

- **EL Procesador.**

Es el elemento que se comunica con los distintos componentes de la CPU mediante los buses de datos, direcciones y control. Su tarea principal consiste en la lectura de las instrucciones del programa de usuario, o de aplicación, y su resolución mediante el empleo de los estados de las entradas y salidas del sistema.

En los autómatas actuales, **el procesador** lo constituyen una o varias placas de circuito impreso, en donde alrededor de un microprocesador se agrupan una serie de chips, principalmente **memorias** e **interfaces**. En la memoria del procesador (ROM del sistema) el fabricante ha grabado una serie de **programas ejecutivos fijos**, firmware o software del sistema y es a estos programas a los que accederá el microprocesador para realizar las funciones ejecutivas que correspondan en función del tiempo en que trabaje.

El software de sistema de cualquier autómata consta de una serie de funciones básicas que realiza en determinados tiempos de cada ciclo: en el inicio o conexión, durante el ciclo o ejecución del programa y a la desconexión.

Este software o programa del sistema es ligeramente variable para cada autómata, pero en general, contiene las siguientes funciones:

- Vigilar que el tiempo de ejecución del programa de usuario no excede el **tiempo de scan** (scan time). A esta función se le suele denominar **watchdog** (perro guardián).
- El **tiempo de scan** es determinado por el lapso de tiempo que existe entre la ejecución por parte del PLC de un ciclo completo. Cuanto más pequeño sea el tiempo de scan, mayor será la velocidad de respuesta del PLC.
- Autotest en la conexión y durante la ejecución del programa.
- Inicio del ciclo de exploración de programa y de la configuración del conjunto.
- Generación del ciclo base de tiempo.
- Comunicación con periféricos y unidad de programación.

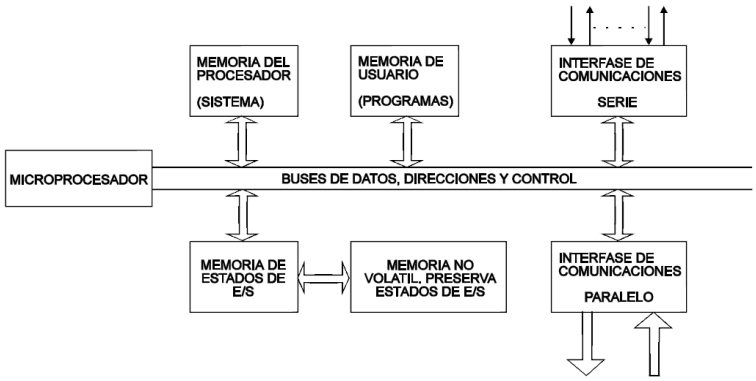

Figura 4.3. Esquema de bloques de la CPU de un autómata basado en microprocesador.

Hasta que el programa del sistema no ha ejecutado todas las acciones necesarias que le corresponden, no se inicia el ciclo de programa de usuario.

Por último es importante resaltar que no todos los autómatas programables utilizan microprocesador como elemento base del procesador (corazón a su vez de la CPU), algunos fabricantes emplean como tal los PLD (Dispositivos Lógicos Programables): FPGA, PAL, etc. También los ASIC (Circuitos Integrados de Aplicación Específica) se utilizan en algunas ocasiones.

❖ La Memoria.

Por otra parte, **la memoria**, otro componente de la CPU, está perfectamente organizada en áreas de trabajo específicas, a diferencia de otros equipos programables.

La memoria del autómata se puede desglosar en secciones, como se ve en la **Figura 4.4**, para entender mejor sus funciones dentro del PLC.

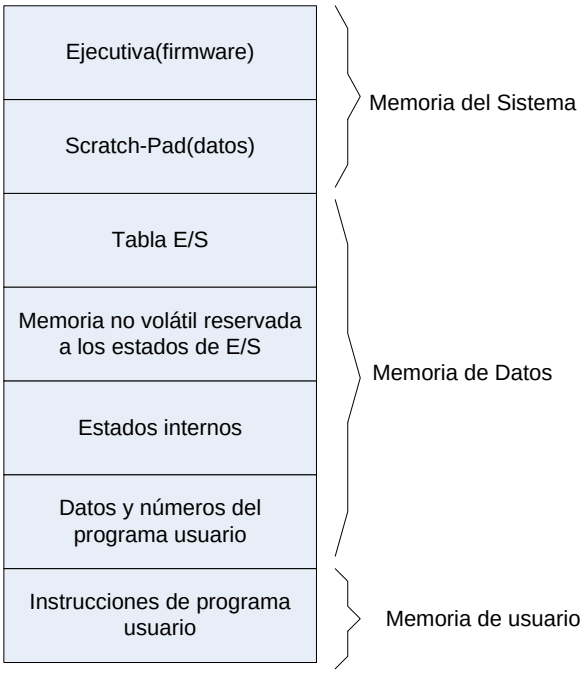

Figura 4.4. Mapa de memoria del autómata.

- **Memoria del Sistema**: también denominada del procesador, es una zona del mapa que generalmente no es accesible por el usuario (por lo menos en su totalidad), en donde se almacenan los **programas ejecutivos** o **firmware**, y un espacio de memoria de almacenamiento temporal intermedio que es empleado por los programas ejecutivos y que suele denominarse **Memoria scratch-pad**.
  La memoria del sistema contiene el software necesario para que el autómata desarrolle su labor como tal.
- **Memoria (de la Tabla) de Datos**: en ella se almacena la información del estado de las E/S (variables de E/S), estados internos intermedios o auxiliares (variables internas) y los datos o números (variables numéricas).
- **Memoria de Usuario**: también llamada de **Programa** o de **Trabajo**, es donde residen las instrucciones que definen el algoritmo de control.

Sería interesante destacar que existe la zona de **memoria no volátil** reservada especialmente para preservar los estados de las E/S en caso de caída de la alimentación o bloqueo accidental del Autómata. Esto nos permitirá recuperar, si se cree oportuno, el estado de las E/S en el proceso de reinicialización del sistema.

Según la tarea a que se dedica, cada una de las **áreas de memoria** puede estar constituida por unidades (chips) de distinta tecnología, aunque a efectos del procesador constituyen un bloque único y ordenado (lineal) al que accede a través del **bus de direcciones.**

Es importante, en todo caso, señalar en las características del autómata la cantidad de memoria disponible para el **programa de usuario**. Así, en el área de **Memoria del Sistema**, el tipo de memoria predominante será del tipo **ROM** en cualquiera de sus versiones (PROM, EPROM, EEPROM, etc.), mientras que en el resto de áreas de la memoria será del tipo **RAM** (volátil o no volátil).

En los Autómatas [7] existe una correspondencia directa entre las direcciones de la **Tabla de E/S** (Memoria de Datos) y los bornes de los módulos o **tarjetas de E/S**, tal como se ilustra en la **Figura 4.5.** Cada fabricante indica cuál es la relación, señalando qué zonas corresponden a entradas y cuales a salidas, aunque en algunos sistemas esta asignación es libre debido a que los módulos se identifican ante el procesador que se encarga de adaptar la **Tabla de E/S** (Memoria de Datos) a la configuración constructiva del **sistema de E/S.**

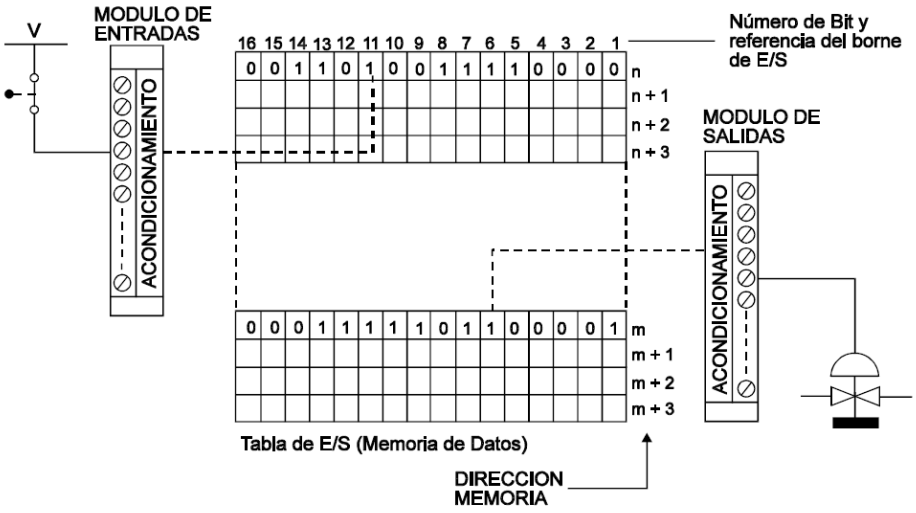

Figura 4.5. La correspondencia entre la Tabla de E/S y los bornes de E/S.

Es muy importante tener presente esta correspondencia entre **la Tabla de E/S** y los **circuitos físicos de E/S** (bornes), ya que en ella se basa la resolución de las instrucciones y la propia programación; al leer las instrucciones, el **procesador** acude a las direcciones de la **Tabla de E/S** para tomar el estado de las variables (bits) asociadas al código de operación. El estado de las variables se almacena en la memoria de datos según el convenio de lógica positiva, es decir, un **"1"** cuando existe un nivel de tensión en el borne del circuito de entrada y un **"0"** en caso contrario.

- **Funcionamiento Cíclico de la CPU.**

Otro aspecto que cabe destacar de los autómatas es el **funcionamiento cíclico de la CPU**, que es una característica original y única de los autómatas programables, que lo diferencia de la mayoría de las maquinas programables cuyos programas tienen normalmente otra estructura que suele finalizar en una instrucción de "**fin de ejecución**". Dicho de otro modo, la filosofía de funcionamiento del autómata es menos flexible que la de, por ejemplo, un ordenador, siendo sin embargo más efectivo que éste en las tareas para las que se ha diseñado.

Se puede explicar el funcionamiento de un Autómata imaginando un cilindro, como el de la **Figura 4.6**, que gire a velocidad constante y que representa la **memoria de programa** (Usuario). Cada una de las 'n' generatrices del cilindro contiene una instrucción. Un **lector** permite acceder a la instrucción para transferirla al Registro de Instrucciones. El tiempo de una rotación completa del cilindro es el **ciclo de ejecución** del Autómata, mientras que el tiempo de paso de una generatriz ante el **lector** es el **tiempo de ejecución** de una instrucción

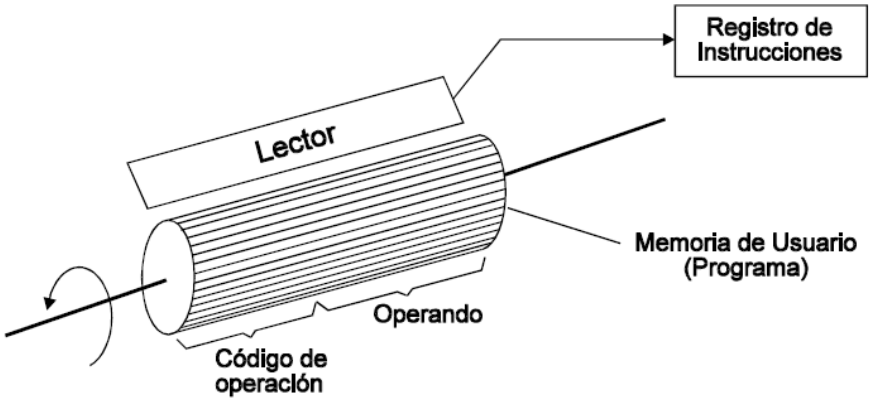

Figura 4.6. Funcionamiento cíclico de la CPU de un Autómata.

En realidad el programa es estático; es el puntero el que, describiendo sucesivamente (salvo en el caso de "salto") todas las palabras de la **memoria de usuario** (programa) hasta la última, antes de recomenzar a partir de la primera, asegura el funcionamiento cíclico (o síncrono) de la CPU.

La potencia de una CPU es directamente función de su velocidad: se llama **período de un Autómata** al tiempo de ejecución de 1K-instrucciones lógicas.

Durante un ciclo, el Autómata asegura las funciones del sistema, es decir, la gestión interna, el autodiagnóstico y el intercambio con el exterior, además, lógicamente, de los tratamientos especificados por el programa. Un **ciclo real** de la máquina comprende dos fases: **la fase-sistema** y la **fase-tratamiento**. El **ciclo de usuario** se limita a esta segunda fase puesto que la primera y, principalmente, las E/S están implícitas y no aparecen.

Teniendo en cuenta que la **fase-sistema** va intercalada a lo largo de la ejecución del programa y que las E/S suponen el 11% (aproximadamente) de la fase-tratamiento, se puede decir que el **tiempo de ejecución** es el que principalmente consume el tiempo total del **ciclo real** del autómata. Una representación de lo comentado se puede verla en la **Figura 4.7.**

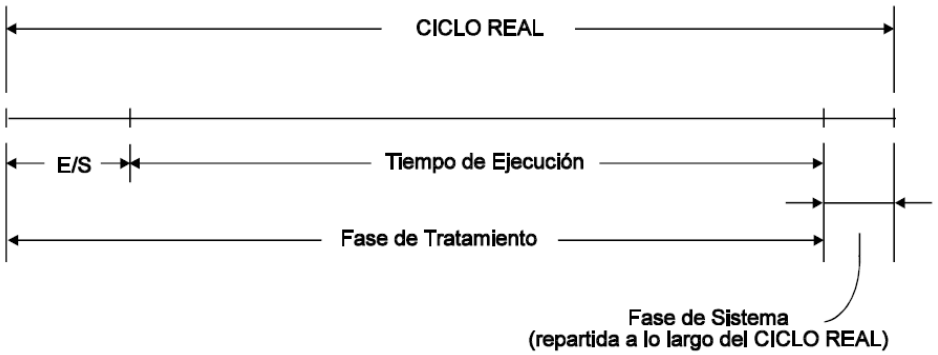

Figura 4.7. Ejemplo del ciclo real de trabajo de un Autómata.

De todo lo anterior podemos deducir que la duración de un ciclo depende fundamentalmente del **número de E/S**, de la **longitud del programa** y de la **velocidad intrínseca del Autómata**.

- **Fuente de alimentación.**

Como antes se ha mencionado, la **fuente de alimentación** puede estar incorporada a la CPU o ser un modulo aparte. Esta adapta la tensión de red a la tensión de alimentación de los circuitos electrónicos internos del PLC y de los dispositivos de E/S.

Algunas fuentes de alimentación proporcionan salidas de auxiliares de tensión con el fin de alimentar a algún sensor, en caso de que fuese necesario, evitando de este modo la necesidad de utilizar una fuente externa adicional.

4.5.2.2 El sistema de entradas y salidas (E/S).

El control efectivo de una máquina o proceso se basa en un continuo intercambio de información entre el equipo de control y dicho proceso. La información que se recoge del proceso recibe el nombre genérico de **Entradas**, mientras que las acciones de control sobre la máquina o proceso se denominan **Salidas**.

Los dispositivos de entrada son los iniciadores de las **señales de entrada** y corresponden a un amplio conjunto de elementos como, interruptores final de carrera, pulsadores, detectores de posición y sensores en general, mientras que los **dispositivos de salida** se encargan de aportar potencia a las señales de salida generadas por el sistema de control y corresponden a relés, contactores, arrancadores de motores, electroválvulas y actuadores en general.

El **sistema de E/S** de un autómata está formado por un conjunto de módulos (o tarjetas) y estructuras de soporte de los módulos o bastidores de montaje, que tiene las siguientes funciones:

- Adaptar la tensión de trabajo de los dispositivos de campo a la de los elementos electrónicos del autómata y viceversa.
- Proporcionar una adecuada separación eléctrica entre los circuitos lógicos y los circuitos de potencia. Este se consigue gracias a un dispositivo electrónico de interfaz, que además de lo anterior, también proporciona una separación galvánica mediante optoacopladores.
- Permitir, mediante el soporte físico del "direccionado" la identificación de los dispositivos de E/S para la correcta ejecución de las secuencias de control programadas.

❖ **E/S Todo-Nada.**

Existen diferentes tipos de señales de **E/S** con la que el autómata tiene que trabajar. Se puede empezar describiendo las **Entradas/Salidas discretas**, en las que se agrupan aquellos componentes destinados a la captación o generación de señales de y hacia dispositivos con dos estados diferenciados, que corresponden a la ausencia o presencia de tensión o corriente, ya sea **CC** o **CA**.

❖ **E/S Numéricas.**

Otro tipo de señales **E/S** son las **Entradas/Salidas numéricas**, que suelen aparecer en autómatas de gama alta o media que efectúan tratamientos numéricos. Los operandos numéricos son generalmente de la misma longitud que la palabra de memoria del Autómata (8, 16 bits). Una tarjeta de **E/S numérica** se presenta pues como un conjunto de entradas o salidas binarias ensambladas.

❖ **E/S Analógicas.**

Un tercer tipo de modulo de **E/S**, son los módulos **analógicos,** Son capaces de captar más de dos estados de una señal. Como el microprocesador trabaja con señales binarias, deberá existir en este módulo un conversor Analógico/Digital, que transforme la señal de entrada en binaria de 10, 11 o 12 bits, dependiendo de las características del conversor utilizado. El nº de bits usado en la conversión es un parámetro importante en un módulo de entradas analógicas pues define la precisión de la lectura. Las señales analógicas de entrada se manejan a nivel de byte o palabra (8 ó16 bits) dentro del programa de usuario.

❖ **E/S de Códigos Numéricos.**

Para la adquisición de datos proporcionados a través de codificadores rotativos o instrumentos electrónicos digitales, y para generar información numérica a dispositivos visualizadores (displays 7 segmentos) y otros equipos electrónicos, los fabricantes de autómatas ofrecen módulos de **E/S de códigos numéricos** generalmente para el código BCD.

❖ **E/S Especiales.**

Por otro lado están las **E/S especiales**. Las **E/S** descritas anteriormente permiten al autómata operar en un amplio campo de aplicaciones, pero algunas de éstas requieren ciertas funciones especiales, que si bien podrían realizarse con los elementos comentados, exigirían un empleo excesivo, tanto de hardware de acondicionamiento como de instrucciones de programa (software).

Para la resolución de estas funciones especiales los fabricantes ofrecen una serie de módulos especializados también denominados "**inteligentes**" o de "**proceso**"; con el empleo de estos módulos se reduce la cantidad de componentes del equipo y se descarga el trabajo del procesador del autómata. Las **E/S especiales** forman parte de los Autómatas de la gama alta y media, aunque progresivamente se van incorporando a los sistemas de gama baja.

Dentro de este tipo de **E/S** se encuentran:

- **Entradas de termopar**: Los módulos analógicos no son capaces de recibir señales de bajo nivel directamente de los transductores y precisan del empleo de convertidores de señal a nivel de instrumentación. Algunos fabricantes ofrecen módulos que aceptan estas señales débiles, del orden de milivoltios y operan como si se tratase de señales analógicas.
- **Controlador de motor paso-paso:** Este modulo genera los trenes de impulso necesarios para el control de un motor paso-paso, a través del amplificador de gobierno (driver). El modulo acepta datos desde el programa de control (CPU) que especifican el recorrido o posición, sentido, aceleración y deceleración del movimiento.
- **Servo-controlador:** Este modulo permite el control de posicionamiento en un eje, o multieje, proporcionando tiempos cortos de posicionado, alta precisión, buena fiabilidad y alta repetitividad, y ofrece una alternativa económica al empleo del control numérico de pequeñas aplicaciones que no requieren las elevadas prestaciones del CNC (Computer Numerical Control).
- **Modulo de control PID:** El objetivo que tiene este modulo es el de poder regular una variable física de un sistema o proceso. Se tendrá que configurar correctamente e introducir los correctos parámetros de control. Este modulo se ha utilizado en este proyecto, En el siguiente capítulo se explicara este modulo con mas detalles ya que es gracias a este que se consigue regular la presión de nuestro deposito.

❖ **Modulos ASCII y Modulos para E/S Remotas.**

En último lugar están los **Módulos ASCII** y los **Módulos para E/S remotas**. El primero permite la conexión de un variado conjunto de periféricos de entrada y salida: monitores, impresoras, lectores de códigos de barras, de bandas magnéticas, teclados, visualizadores, etc.

EL segundo tipo de modulo se utiliza en sistemas de control de gran tamaño donde no es posible ubicar todos los módulos de entradas en el mismo rack, siendo necesaria una expansión. Existen dos posibilidades: la primera es un enlace paralelo mediante un cable multiconductor (bus). Esta solución no es muy recomendable ya que al extender la longitud del cable se producen caídas óhmicas de tensión y además el sistema se vuelve más sensible a interferencias electromagnéticas. La segunda opción es establecer una conexión de las entradas remotas mediante un módulo de interconexión serie del tipo **bus de campo**, lo que permite conectar el PLC con el sistema remoto, mediante un par de hilos, logrando una descentralización de las entradas que puede llegar al orden de los kilómetros.

4.5.2.3. Equipo de programación y periféricos.

Los equipos de programación y periféricos [7] son componentes, en algunos casos imprescindibles, del entorno del Autómata Programable que facilitan, en general, la comunicación de éste con otros equipos o personas (operarios y técnicos).

La elección de la configuración del autómata, es decir, la elección de dispositivos internos (procesadores, etc.) y externos (periféricos, equipos de programación) le complementan formando un sistema apto para ejecutar o desarrollar las misiones que le son asignadas.

❖ **La Consola de Programación.**

En primer lugar tenemos a la **consola de programación** es el medio material del que se auxilia el programador para grabar o introducir en la **memoria de usuario** las instrucciones del programa, en otras palabras, permite comunicar al usuario con el autómata.

Sus funciones básicas son las siguientes:

- Transferencia y modificación de programas.
- Verificación de la programación.
- Información del funcionamiento de los procesos y estados y funcionamiento de los elementos de E/S.
- Control de operación.

La programación del autómata se suele realizar empleando alguno de los siguientes elementos:

- **Unidad de programación:** suele tener la apariencia de una calculadora. Es la forma más simple de programar el autómata, y se suele reservar para pequeñas modificaciones del programa o la lectura de datos en el mismo lugar en que está instalado el autómata.

- **PC:** es el modo más potente y el más utilizado en la actualidad. Permite programar desde un ordenador personal estándar, con todo lo que ello supone: herramientas más potentes, posibilidad de almacenamiento en soporte magnético, impresión, transferencia de datos, monitorización...

Para cada caso el fabricante proporciona lo necesario, bien el equipo o el software y los cables adecuados. Cada equipo, dependiendo del modelo y fabricante, puede poseer una conexión a uno o varios de los elementos anteriores.

❖ **Los Periféricos.**

Por otra parte tenemos **los periféricos**, que gracias a la conexión con interfaces se puede ampliar en tamaño y prestaciones al autómata. Las ampliaciones abarcan muchas posibilidades, que van desde las redes internas (LAN, etc.), módulos auxiliares de E/S, memoria adicional, incluso la conexión con otros autómatas del mismo modelo.

Cada fabricante facilita las posibilidades de ampliación de sus modelos, los cuales pueden variar incluso entre modelos de la misma serie.

Los periféricos no intervienen directamente en el funcionamiento del autómata, pero sin embargo facilitan la labor del operario. Sirven para documentar los programas y supervisar el proceso. Los más utilizados son: impresoras, cartuchos de memoria EEPROM, visualizadores, monitores y paneles de operación OP, unidades de disco, fax, LEDs y teclados. [7]

4.6. El Autómata Simatic S7-300.

El sistema de automatización **SIMATIC S7-300** es un sistema modular de control para sistemas medianos y pequeños. Permite una adaptación optima en las tareas a automatizar ya que existe la posibilidad de una ampliación en los módulos (entradas/salidas digitales, entradas/salidas analógicas, entre otras).

4.6.1. Elementos de manejo y visualización de la CPU 314C 2 DP.

La **CPU 314 2 DP** (nº ref. 6ES7314-6CG03-0AB0) del sistema de automatización SIMATIC S7-300 [8], sus entradas y salidas analógicas y digitales, así como algunos elementos importantes se pueden ver en la **Figura 4.8**:

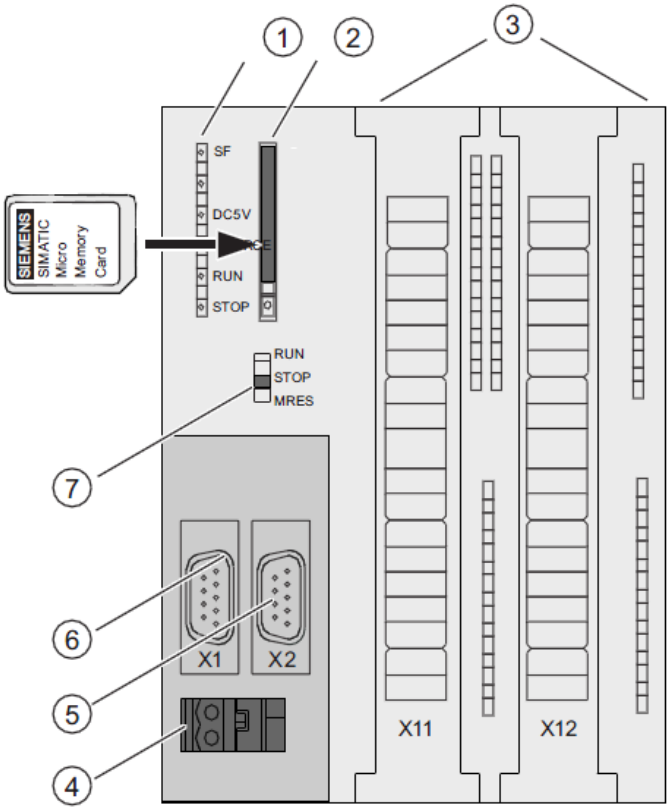

Figura 4.8. Elementos importantes en el PLC.

1 Indicadores de estado y error.

- SF (rojo) Indicador de error de Hardware o de Software.
- BF (rojo) Error de bus.
- DC5V (verde) Alimentación de 5 voltios para CPU y para el bus S7-300, correcta.
- FRCE (amarillo) Petición de forzado permanente activo.
- RUN (verde) CPU en estado Run.
- STOP (amarillo) CPU en estado Stop.

2 Ranura de Micro memory Card con expulsor.

- El módulo de memoria empleado es una Micro Memory Card SIMATIC. Dicho módulo se puede utilizar como memoria de carga o como soporte de datos de bolsillo.
- Puesto que estas CPUs no disponen de memoria de carga integrada, para su funcionamiento es imprescindible insertar una Micro Memory Card SIMATIC.

3 Conexiones de las entradas y salidas integradas.

- Este autómata lleva integrado:

  5 Entradas analógicas y 2 salidas analógicas (configurables en tensión o intensidad).

  8 Entradas digitales en cada grupo.

8 Salidas digitales en cada grupo.

8 Entradas de alarmas en cada grupo.

3 Contadores de alta velocidad.

1 Canal para posicionamiento (al utilizar una función de posicionamiento, sólo se dispone de los canales de contaje 2 y 3).

- La numeración de las entradas y de las salidas es configurable. Se puede cambiar su numeración.

La **Figura 4.9** muestra las entradas y salidas digitales y analógicas integradas de la CPU con las puertas frontales abiertas.

- La cifra 1 corresponde a las entradas y salidas analógicas.
- En la cifra 2 están las ocho entradas digitales en cada grupo.
- Por último está la cifra 3 que se refiere a las 8 salidas digitales en cada grupo.

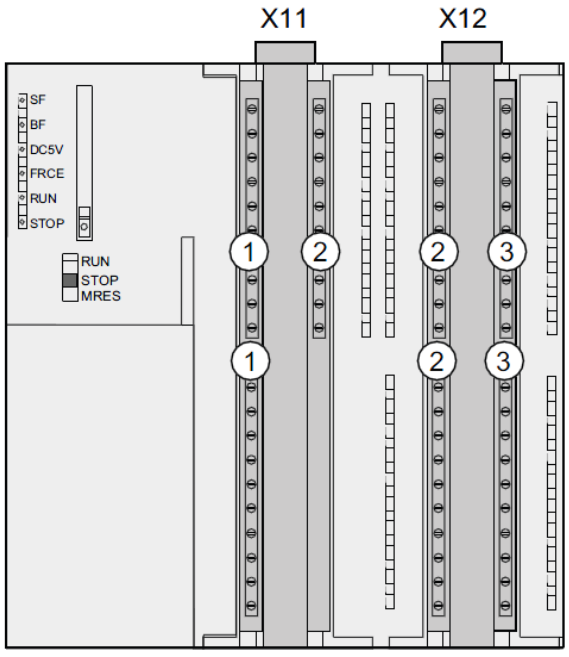

Figura 4.9. Entradas y salidas analógicas y digitales.

4 Conexión para la fuente de alimentación.

- Cada CPU dispone de un conector hembra de 2 polos para la conexión a la fuente de alimentación a 24 V DC que normalmente va enganchada al rack. En estado de suministro, el conector ya está enchufado al conector hembra con bornes de tornillo.

5 Interfaz X2 (PROFIBUS DP).

- La interfaz PROFIBUS DP sirve principalmente para conectar aparatos de la periferia descentralizada. Por ejemplo, con PROFIBUS DP se pueden configurar subredes de gran tamaño.
- La interfaz PROFIBUS DP se puede configurar como maestro o como esclavo, permitiendo utilizar una velocidad de transferencia máxima de 12 Mbits/s.
- Vía PROFIBUS DP podemos conectar aparatos PG/PC, OP/TP, Esclavos o Maestros DP, Actuadores/Sensores o con S7-300/S7-400 con este interfaz.

6 Interfaz X1 (MPI).

- La MPI (Multi Point Interface) es el enlace entre la CPU del PLC y el ordenador o para comunicar una red MPI.
- La velocidad de transferencia predeterminada es de 187,5 Kbits/s.
- Se pueden conectar aparatos vía MPI tales como PG/PC, OP/TP, S7-300/S7-400 con este interfaz y con S7-200 (solo a 19,2 Kbits/s)

7 Selector de modo de operación.

- RUN: CPU en modo RUN, es decir que procesa el programa usuario.
- STOP: Modo de operación STOP, es decir que la CPU no procesa ningún programa de usuario.
- MRES: Borrado total. Es una posición no enclavable del selector de modo para el borrado total de la CPU. El borrado total mediante el selector de modo de operación requiere una secuencia especial de operación.

4.6.2. Concepto de memoria en la CPU 314C 2 DP.

Como se puede observar en la **Figura 4.4**, la memoria de un autómata, en general, se puede dividir en tres partes: **Memoria de Sistema**, **Memoria de datos** y **Memoria de Usuario.** Como se puede observar en la **Figura 4.10**, la memoria de la CPU 314 2 DP [8] se divide de una forma muy similar.

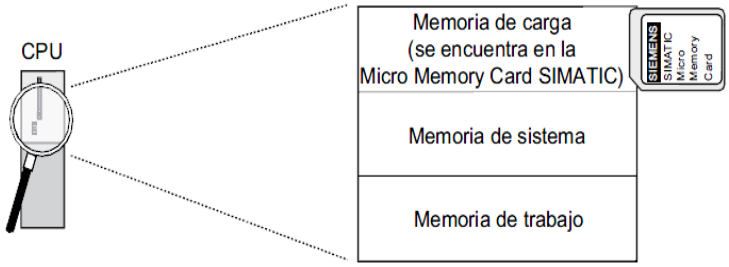

Figura 4.10. Áreas de memoria de la CPU 314C 2 DP.

Aunque las dos graficas de las áreas de memoria sean similares, es importante destacar lo siguiente:

❖ **Memoria de carga.**

La memoria de carga se encuentra en la Micro Memory Card SIMATIC (máx. 8MB) y equivale exactamente al tamaño de esta. La memoria de carga contiene los objetos generados por la unidad de programación (programa usuario, es decir, todos los bloques, OBs, FCs, FBs, DBs, la información y la configuración del sistema). Para utilizar la CPU es imprescindible tener una MMC.

La memoria de carga siempre es remanente: Se almacena ya durante la carga en la Micro Memory Card SIMATIC de forma protegida contra cortes de alimentación y un borrado total.

❖ **Memoria de sistema.**

La memoria de sistema está integrada en la CPU y no se puede ampliar. Diseñada como memoria RAM, guarda las áreas de operandos (**Tabla 4.2**), así como las áreas de datos requeridas internamente por el sistema operativo (p.ej. búfer para la comunicación).

La remanencia de la memoria de sistema puede ser configurable para el caso de marcas, temporizadores y contadores, es decir, que partes deben ser remanentes y cuáles deberán inicializarse a “0”. Para el caso del búfer de diagnostico, la dirección MPI (y su velocidad de transferencia) así como el contador de horas de funcionamiento suelen almacenarse en el área de memoria remanente de la CPU.

❖ **Memoria de trabajo.**

La memoria de trabajo está integrada en la CPU (96KB) y no se puede ampliar. Sirve para procesar el código y los datos del programa de usuario. Este procesamiento tiene lugar exclusivamente en el área de la memoria de trabajo y en la memoria del sistema.

La memoria de trabajo contiene bloques de datos (DB) remanentes y no remanentes. Los bloques de datos remanentes se pueden cargar en la memoria de trabajo hasta el límite de remanencia máximo de dicha memoria. El tamaño máximo de la memoria para estos bloques de datos remanentes es de 64KB.

Por lo tanto, el programa usuario tendrá un consumo de **memoria de carga** y otro de **memoria de trabajo**. No se deberá superar la cantidad de **memoria de trabajo** disponible, ya que no es posible su ampliación, por lo que se tendría que cambiar la CPU. La **memoria de carga** sí que se podría ampliar mediante Flash o RAM externas.

4.6.2.1. Área de operandos en la memoria de sistema.

La memoria de sistema de la CPU 314 DP está dividida en áreas de operandos. Utilizando determinadas operaciones, el usuario direcciona los datos en su programa directamente en el área de operandos que corresponda.

Área de operandos	Descripción
Imagen de proceso de las entradas	Al comienzo de cada ciclo OB 1, la CPU lee las entradas de los módulos de entrada y guarda los valores en la imagen de proceso de las entradas.
Imagen de proceso de las salidas	Durante el ciclo, el programa calcula los valores de las salidas y los deposita en la imagen de proceso de las salidas. Al final del ciclo OB 1, la CPU escribe los valores de salida calculados en los módulos de salida.
Marcas	Este área dispone de espacio en memoria para los resultados intermedios del programa.
Temporizadores	En este área residen los temporizadores.
Contadores	En este área residen los contadores.
Datos locales	Este área de memoria recoge los datos temporales de un bloque lógico (OB, FB, FC) mientras dura el procesamiento de este bloque.
Bloques de datos	Los bloques de datos (DB) son áreas de datos en el programa de aplicación que contienen datos del usuario.

Tabla 4.2. Área de operandos de la memoria de sistema.

Cabe destacar de la **Tabla 4.2** la zona de la memoria de sistema que se llama imagen de proceso. Cuando no se consulta el estado de las señales en los **módulos de señales**, sino que se accede a un área de la memoria de sistema de la CPU, se está accediendo a la **imagen de proceso**. Esta imagen de proceso está dividida en dos partes: la imagen de la entrada y la de las salidas.

El acceso a la imagen de proceso presenta, frente al acceso directo a los módulos de entrada y salida, la ventaja de que la CPU ofrece una imagen consistente de las señales de proceso durante la ejecución cíclica del programa. No se puede dar el caso de que en una subrutina una señal tenga un valor, y microsegundos después tenga otro.

El estado de la imagen de proceso, como se puede observar en la **Figura 4.11**, no cambiará o no se actualizará hasta el siguiente ciclo. Además, el acceso a la imagen de proceso requiere menos tiempo que el acceso directo a los módulos de señal, ya que la imagen del proceso se encuentra en la memoria de sistema de la CPU.

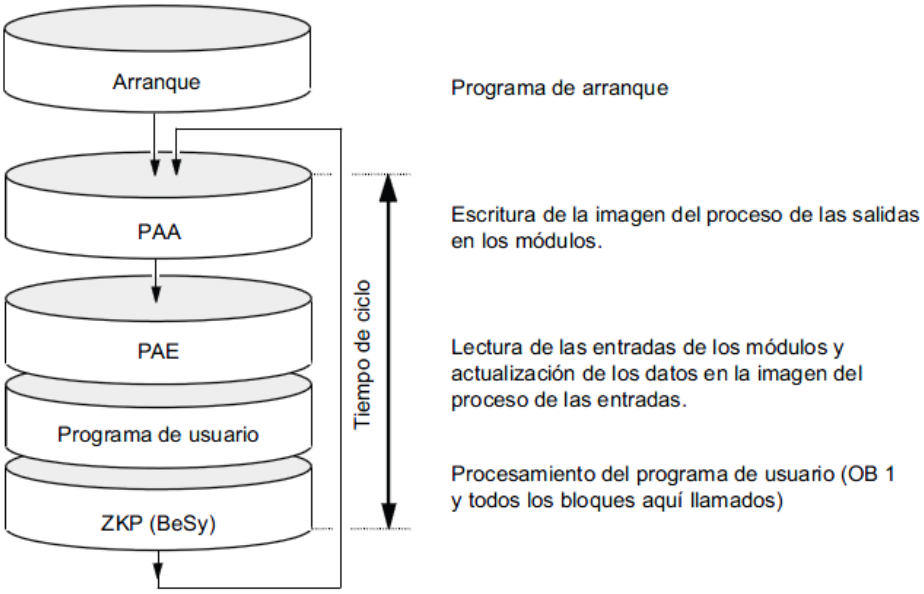

Figura 4.11. Proceso de actualización de la imagen de proceso.

4.6.2.2. Tiempo de ciclo.

El tiempo de ciclo (**Figura 4.11**) [8] es el tiempo que necesita el sistema operativo para procesar un ciclo del programa, es decir un ciclo de OB 1, así como todos los componentes y actividades del sistema que interrumpen dicho ciclo. Este tiempo se supervisa, ya que esta ejecución cíclica del programa y, con ello, la del programa de usuario se realiza en segmentos de tiempos definidos.

Cada CPU SIMATIC, tiene valores de tiempo predefinidos para cada byte o proceso interno de la CPU. Por ejemplo se podría calcular fácilmente el tiempo en que se tardea en actualizar la imagen de proceso de cada ciclo si nos fijamos en estos tiempos predefinidos, así como también el **tiempo de ciclo** o el **tiempo de respuesta** del PLC.

Los tiempos indicados pueden prolongarse por diversos motivos, entre ellos se destacan: la ejecución de alarmas horarias, tratamiento de erros o debido a la comunicación de la CPU, etc.

4.6.3. Fuente de alimentación PS 307; 2A.

Para alimentar el S7-300 y a los sensores/actuadores con 24 V c.c. se dispone de la fuente o modulo PS 307; 2A (nº ref. 6ES7307-1BA00-0AA0), cuyas propiedades están definidas en el capítulo 3, en la sección 3.3.

Esta fuente de alimentación está ubicada a la izquierda de la CPU (**Figura 4.8**) y presenta la forma de la **Figura 4.12.**

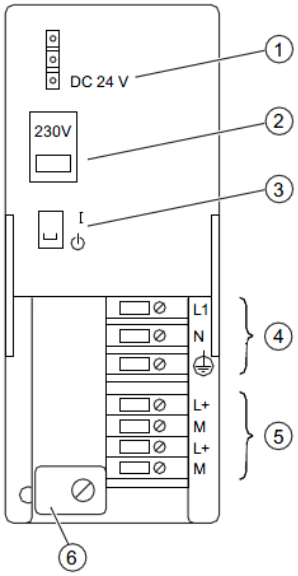

Figura 4.12. Fuente de alimentación PS 307; 2A.

1 Indicador de “Tensión de salida 24 V DC aplicada”.

2 Selector de tensión de red.

3 Interruptor On/Off para 24 V DC.

4 Bornes para tensión de red y conductor de protección.

5 Bornes para tensión de salida 24 V DC.

6 Alivio de tracción.

4.6.4. Periferia integrada.

Para la CPU 314C 2 DP [8] las **salidas** y **entradas**, ya sean **analógicas** o **digitales**, se encuentran integradas en el autómata. Estas E/S integradas se pueden utilizar de dos formas; como periferia estándar o para funciones tecnológicas, es decir, para alarmas, contaje, medición de frecuencia y modulación de ancho de pulso. En este caso todas la E/S utilizadas son de forma estándar. Las periferias que están integradas en esta CPU son:

- La periferia digital integrada DI 16/DO16 x DC 24
- La periferia analógica/digital AI 5/AO 2 x 12 Bit /DI8 x DC 24

En este autómata se dispone de 24 entradas digitales parametrizables individualmente con separación galvánica, de las cuales 16 se pueden utilizar para las funciones tecnológicas. Estas entradas utilizan una tensión nominal 24Vdc y un tiempo de filtrado nominal de 3ms que se puede modificar.

También se dispone de 16 salidas digitales con separación galvánica, de las cuales 4 son para funciones tecnológicas y además, estas son salidas digitales rápidas. En este caso también utilizan 24Vdc nominales, con una frecuencia de conmutación para las salidas estándar de 100Hz y de 2,5KHz para las salidas rápidas. De las entradas digitales únicamente se utilizará una, un bit, como '**RESET**' de la función PID integrada, esta es la entrada del entrenador maestro/esclavo **E124.7.**

En cuanto a la periferia analógica, se dispone en total de 4 entradas analógicas de tensión e intensidad y 1 entrada de resistencia. Por otra parte, hay 2 salida de tensión e intensidad, las dos con una resolución de 15bits mas signo. En este proyecto se ha utilizado la entrada **PEW 754**, en forma unipolar, para leer la señal del **sensor de presión** y como salida la **PAW 754**, también en forma unipolar, por la cual se emite la **señal de control.**

Esta periferia analógica se puede configurar para que realice medidas de tensión, intensidad, de resistencias y de termopares, siendo las E/S más usadas son las de voltaje e intensidad:

- **Medidas de tensión**: Se puede configurar de forma que mida de 0-10Voltios o de -10 a 10Voltios. La primera configuración, la unipolar, es la más utilizada. El inconveniente de este tipo de lectura es que al ser una tensión, las distancias sin atenuación de la señal, debido a caídas de tensión en el cable, son relativamente cortas.
- **Medidas de intensidad:** Dentro de las medidas de intensidad se suelen gastar principalmente dos tipos: 0- 20 mA y 4-20 mA. Este último tipo es el más utilizado en la lectura analógica, ya que permite grandes distancias al ser la lectura por corriente, y a la vez es fácil reconocer la rotura del hilo, ya que por debajo de 4 mA es que no llega lectura del elemento.

- **Medidas de termorresistencias y resistencias**: Al medir la resistencia, el módulo suministra una corriente constante a través de los bornes de la entrada. La corriente constante se conduce a través de la resistencia a medir. Dicha corriente se mide luego como caída de tensión. Es importante que los conductores de corriente constante conectados se enlacen directamente con la termorresistencia/resistencia.

Las entradas analógicas de los S7 300 se agrupan en grupos de 2 canales, por lo que ambos deben de estar configurados de la misma manera, ya sea para 0-10 V, para 4-20 mA, etc. Por lo tanto, no se puede mezclar en el canal 0 una señal de 0-10 V y en el canal 1 una de 4-20 mA. Deberá de dejarse vacío dicho canal si no se dispone de ninguna sonda de 0-10 V, y cablear la de 4-20 mA en el siguiente grupo libre.

La CPU solo puede procesar los valores analógicos en forma binaria, por tanto, la periferia analógica integrada será la encargada de convertir la señal analógica en una digital, para el caso de las entradas y para las salidas el proceso inverso, de señal digital a señal analógica. Por tanto un valor analógico digitalizado de un mismo rango nominal es idéntico tanto si se trata de un valor de entrada como de salida. Los valores analógicos se representan como cifra de coma fija en forma de complemento de 2. De ello resulta la correspondencia de la **Tabla 4.3**.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Valor del bit	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Tabla 4.3. Representación de los valores analógicos con resolución de 16 bits.

El donde el signo ira en el bit 15, siendo 0 para los positivos y 1 para los negativos. En caso de este documento el rango de entradas y salidas unipolares es el de la **Tabla 4.4.**

Sistema		Rango de medición de tensión		
dec.	hex.	de 1 a 5 V	de 0 a 10 V	
32767	7FFF	5,741 V	11,852 V	Rebase por exceso
32512	7F00			
32511	7EFF	5,704 V	11,759 V	Margen de saturación
27649	6C01			
27648	6C00	5 V	10 V	Rango nominal
20736	5100	4 V	7,5 V	
1	1	1 V + 144,7 µV	0 V + 361,7 µV	
0	0	1 V	0 V	
-1	FFFF		valores negativos imposibles	Margen de saturación por defecto
-4864	ED00	0,296 V		
-4865	ECFF			Rebase por defecto
-32768	8000			

Tabla 4.4. Representación de valores analógicos.

Para transformar este valor analógico a uno digital, la CPU utiliza el principio de medida de la codificación momentánea o de aproximaciones sucesivas, que es uno de los procedimientos más sencillos e importantes a la hora de codificar.

Para aplicar este método trabaja con un coeficiente de exploración de 1kHz; es decir, cada milisegundo aparece un nuevo valor en el registro de la entrada de la periferia y se guarda como imagen de proceso, donde puede leerse en el programa usuario. Si el tiempo de acceso a la periferia es menor a 1 ms se vuelve a leer el mismo valor. Al registrar la entrada, se garantiza que la CPU disponga de una imagen cogerente de las señales del proceso durante la ejecución cíclica del programa.

4.7. Entorno de programación SIMATIC MANAGER.

En los autómatas programables las tareas de automatización se formulan mediante programas. En ellos, el usuario fija en una serie de instrucciones para indicar cómo el autómata debe controlar un proceso o realizar una tarea determinada. Para que el autómata pueda "entender" el programa, éste debe estar escrito ciñéndose a unas reglas prefijadas y escribiendo en un lenguaje determinado (**el lenguaje de programación)**. Para la familia SIMATIC S7 se desarrolló el lenguaje de programación **Simatic Manager STEP 7**. Más concretamente, en este proyecto se utilizo la versión V5.4 de la herramienta de programación **Simatic Manager**.

Simatic Manager es un entorno profesional flexible que permite programar un gran número de autómatas de la familia SIEMENS a través de un PC. Para ello está dotado de una potente colección de librerías que almacenan las características específicas HW (hardware) de cada autómata y los elementos de conexionado para configuraciones en red. Un proyecto con esta herramienta de programación abarca toda la gestión de programas y datos de una solución de automatización, independientemente del número de módulos y de cómo estén interconectados. Por consiguiente, el proyecto no se limita solamente a un programa de usuario destinado a un modulo programable, sino que puede englobar varios programas de usuario para varios módulos programables que se encuentren bajo un mismo nombre de proyecto.

4.7.1. Tipos de módulos de memoria.

Simatic Manager S7 dispone de una serie de módulos [9] que dividen la memoria de programa, que dependiendo del modelo en concreto, varían en número y funcionalidad. Esta división permite una programación estructurada y un acceso ordenado a los datos.

Estos módulos se programan en la herramienta Simatic Manager, este es el **programa usuario**. Este se transfiere completamente desde el PC a la CPU mediante la Micro Memory Card SIMATIC. Al hacerlo, se borra el contenido anterior de la Micro Memory Card. En la memoria de carga, todos estos bloques ocuparan un lugar previamente definido. Este proceso se puede apreciar en la **Figura 4.13.**

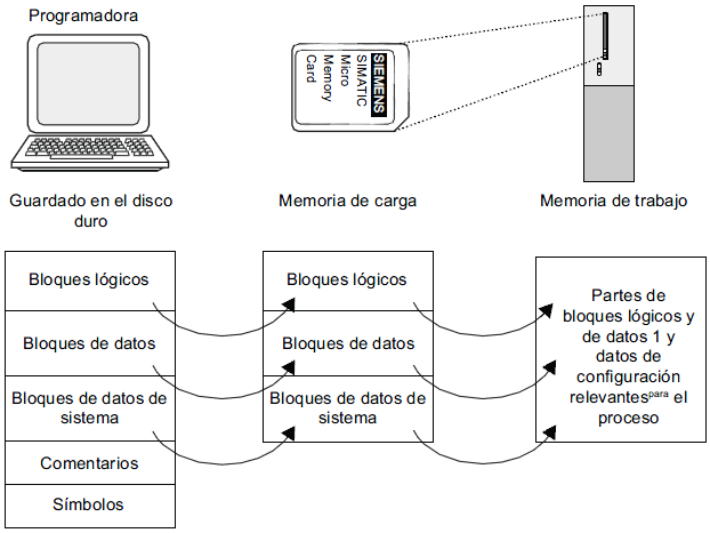

Figura 4.13. Carga del programa usuario.

❖ Bloques de Organización (OB).

Los bloques de organización (OBs) constituyen el interfaz entre el sistema operativo y el programa de usuario. Los distintos bloques de organización se reparten diferentes tareas. El OB1 es la subrutina principal en la que empieza el ciclo de programa de PLC y por la que finaliza. Toda subrutina o bloque para poder ser ejecutada debe ser llama desde el OB 1. De no ser así no se ejecutará, pese a encontrarse en la memoria del PLC.

Además de la el OB1 existen muchas más, cada una con una función específica y prioridad sobre las demás, entre otras, se pueden ver las más importantes en la **Tabla 4.5.**

	FUNCIÓN	S7
Programa principal	Ciclo libre	OB1
Alarma	Alarma de retardo	OB20 a OB23
	Alarma horaria	OB10 a OB17
	Interrupciones de hardware	OB40 a OB47
	Alarmas cíclicas	OB30 a OB38
	Alarmas multiprocesamiento	OB60
Arranque	Nuevo arranque manual	OB100
	Rearranque manual y automático	OB101
Error	Error	OB 121, OB122, OB80 a OB87
Otras	Tarea no prioritaria	OB90

Tabla 4.5. Módulos de memoria.

Se podrían dedicar algunas páginas para explicar las diferentes OBs existentes, pero para este proyecto solo importa destacar las OBs de alarma cíclica. Estas son bloques que se pueden llamar a intervalos regulares de tiempo por el sistema operativo del PLC. Para que se produzca la llamada a las mismas el único requisito es su existencia dentro de la memoria del PLC. El periodo de tiempo para las llamadas a estas OBs es parametrizable al programar el autómata.

En los Simatic S7 300 [9] únicamente se encuentra habilitada la OB35 para estos menesteres. La OB35 se suele utilizar para las llamadas a lecturas analógicas, regulación PID y otros procesos que requieran un tratamiento uniforme en el tiempo, e independientemente del tiempo de ciclo del programa. Debido a las anteriores características esta OB tiene un papel importante en este documento.

❖ **Función (FC).**

Una función (**FC**) es un bloque lógico "sin memoria". Los parámetros de salida contienen los valores que se obtienen tras ejecutar la FC. Estos bloques tienen como identificador las letras **FC<**Número**>** y sirven para descomponer el programa en elementos funcionales o tecnológicos, facilitando la depuración y el mantenimiento.

❖ **Bloques de Función (FB).**

Un bloque de función (**FB**) es un bloque lógico "con memoria". Como memoria se utiliza un bloque de datos de instancia que sirve para almacenar los parámetros actuales y los datos estáticos de bloques de función.

Permiten encapsular elementos más complejos, teniendo como identificador **FB**<Numero>. Comparados con los bloques **FC**, presentan la ventaja de que pueden programarse con un juego de instrucciones ampliado y tienen que estar asociados a un bloque de datos (**DB**<Número>).

❖ **Bloque de Datos (DB).**

Son áreas de memoria destinadas a contener datos del programa de usuario. Estos (**DB**<Número>) pueden estar o no asociados a bloques FB. Si están asociados se llaman **módulos de datos de instancia** y solo pueden ser utilizadas desde dicha **FB** y si no lo están, se llaman **módulos de datos globales** y pueden ser utilizados por cualquier bloque o modulo del programa.

❖ **Bloques de Sistema.**

No todas las funciones tienen que ser programadas por el usuario. Existen bloques preconfeccionados que residen en el sistema operativo de los módulos centrales. En particular se trata de los bloques siguientes:

- **Funciones de sistema (SFC):** con las características de una función FC.

- **Bloques de función de sistema (SFB):** con las características de un bloque de función FB.
- **Bloques de datos de sistema (SDB):** estas no son funciones sino que contienen los ajustes, como pueden ser los parámetros de módulos o también direcciones.

4.7.2. Lenguajes de programación.

El mismo lenguaje de programación STEP 7 puede representarse de distintas formas [6], lo cual redunda en una mayor comodidad para el usuario. Las distintas formas son el AWL, FUP, KOP. Además el STEP 7 contiene el S7-GRAPH, que es un potente lenguaje grafico que permite programar directamente usando Grafcets.

Los usuarios familiarizados con la programación de microprocesadores probablemente prefieran utilizar el formato **AWL,** ya que este se presenta como una sucesión de abreviaturas (nemónicos) de instrucciones tipo microprocesador. Esta es la forma más potente de programar.

Los usuarios familiarizados con los esquemas de puertas lógicas, biestables, etc., pueden encontrar más cómodo programar en **FUP.** En **FUP** el programa se representa gráficamente con símbolos normalizados procedentes de esquemas de tipo electrónico y que STEP 7 proporciona en forma de librería de funciones.

A los usuarios procedentes del campo de la electricidad les resultará más sencillo expresar los programas en formato **KOP**. En **KOP** se representa el programa gráficamente con símbolos normalizados procedentes de esquemas tipo eléctrico.

Cada forma de representación tiene sus particularidades. Por ello, un módulo de programa que haya sido escrito en **AWL** no puede (en general) trasladarse automáticamente a formato **FUP** o **KOP**. Las formas de representación gráficas tampoco son compatibles entre sí. Sin embargo, siempre es posible traducir a **AWL** los programas escritos en **FUP** o **KOP**. En la **Figura 4.14**, se puede ver claramente lo anterior.

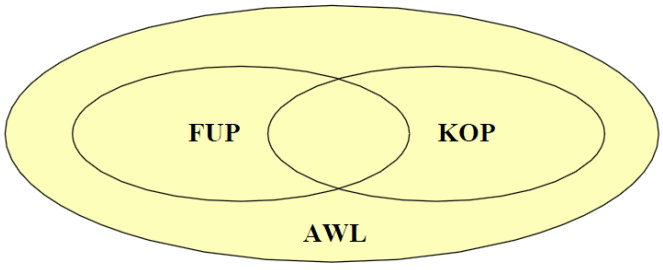

Figura 4.14. Esquema de las tres formas de representación del lenguaje STEP 7

La programación no obliga a utilizar un lenguaje común para todo el proyecto, siendo nuestra propia limitación a la hora de cambiar de idioma lo que nos obliga a obcecarnos en uno en particular, enmascarando sus defectos cuando utilizamos el lenguaje que no corresponde en una tarea de automatización. Por tanto, la decisión no es pues **qué** lenguaje de programación utilizar, sino **cuando** utilizar cada uno.

4.7.2.1. Características de cada lenguaje de programación.

Para conocer dicha respuesta, deberemos aprender primero las características de cada uno de ellos:

❖ **AWL.**

Este lenguaje es el que necesita menos instrucciones de programación de todos, con lo cual el programa ocupa igualmente menos código compilado.

Cuando se requiera realizar grandes operaciones con señales analógicas, el **AWL** es el indicado, aunque los otros dos lenguajes también pueden realizar operaciones con analógicas perfectamente, pero se necesitaría mucho más espacio.

Como permite introducir una gran cantidad de sentencias en la misma pantalla, esto puede ser contraproducente debido a que el programa se convierte en ininteligible hasta para el propio programador.

❖ **KOP.**

Es muy sencillo de ver en él los pasos del programa, y seguir las condiciones del proceso. Es un tipo de lenguaje totalmente indicado para programadores más cercanos al mundo eléctrico que al informático.

Como son símbolos gráficos normalizados, estos requieren mucho espacio y enseguida se salen de la pantalla, por lo que se debe desplazarse muy a menudo. Aunque como solución a esto se podrían utilizar marcas.

Como inconveniente, las cajas de **KOP** necesitan una sistemática de proceso por parte del STEP 7 que hace que no se optimice el código de las mismas, por lo que el programa haciendo lo mismo va más lento.

❖ **FUP.**

Permite realizar gran cantidad de series y paralelos en la misma pantalla, pero con mayor claridad de diagnostico que el **AWL**. Este tipo de lenguaje es el más indicado para los electrónicos.

Presenta los mismos inconvenientes que el **KOP**; no optimiza el código de las mismas, por lo que el proceso es más lento.

Se puede ver un ejemplo de los tres tipos de lenguaje para el mismo esquema eléctrico en la **Figura 4.15.**

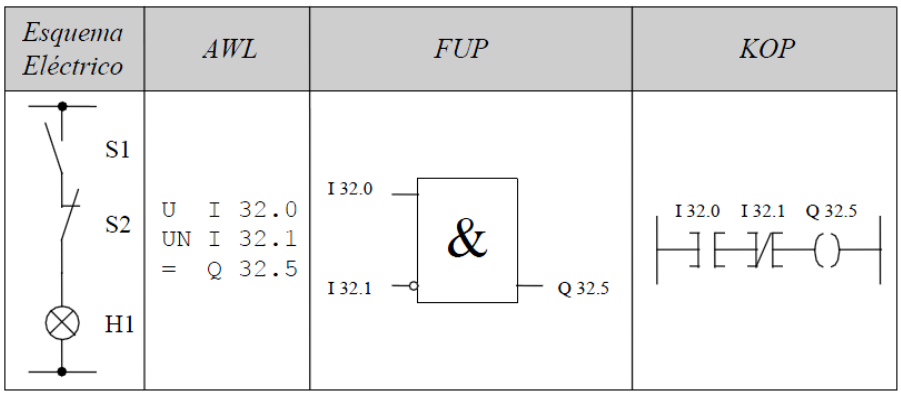

Figura 4.15. Ejemplo de programación en los tres lenguajes.

La respuesta es evidente, cada vez uno. Para las tareas que no sean setear, resetear o activar bits, el **AWL** es sin dudas el lenguaje a utilizar. Las ventajas del mismo sobrepasan ampliamente los inconvenientes.

Sin embargo para todas las activaciones (series y contactos) la decisión debe de ser **KOP** o **FUP**. Aunque también se encuentran los gustos de cada programador que será el que al final decida que lenguaje utilizar.

5. CONFIGURACIÓN Y PROGRAMACIÓN DEL AUTÓMATA PROGRAMABLE.

5.1. Introducción.

En este capítulo se detallará de una manera ilustrativa y fácilmente comprensible cada uno de los pasos que se han seguido para configurar y programar el PLC utilizando el programa **Simatic Manager STEP 7.**

El presente proyecto se puede llevar a cabo gracias al modulo PID que viene preconfigurado en el PLC (**FB 41 CONT_C**). Este modulo PID funciona gracias a un algoritmo de control muy utilizado en la industria de procesos. Este algoritmo PID consiste en aplicar al sistema una señal de control que es la suma de tres términos: término proporcional, derivativo e integral.

El 95% de los bucles de control en la industria son del tipo PID, y fundamentalmente PI. La amplia implantación del control PID en la industria se debe fundamentalmente a que se obtienen resultados satisfactorios para una amplia gama de procesos a pesar de la existencia de perturbaciones sobre el sistema. Estos buenos resultados se pueden obtener sin poseer un gran conocimiento de teoría de control, ya que existen reglas que permiten obtener los parámetros del controlador PID, también cabe destacar que muchos PLC's industriales proporcionan ciertas opciones de autosintonia.

Es por esto que se dedica parte de este capítulo a explicar lo que significan las siglas PID, para luego explicar cómo funciona internamente en el PLC. Pero, en primer lugar, se empieza con una explicación del funcionamiento de todo el sistema.

5.2. Funcionamiento general del sistema.

Hasta ahora se han ido describiendo todos los elementos de este proyecto, a algunos como el PLC se le han dedicado más páginas debido a su relevancia. Es importante saber el funcionamiento de cada elemento por separado, pero también lo es la forma de interactuar de estos, y de este modo poder comprender el funcionamiento global del sistema.

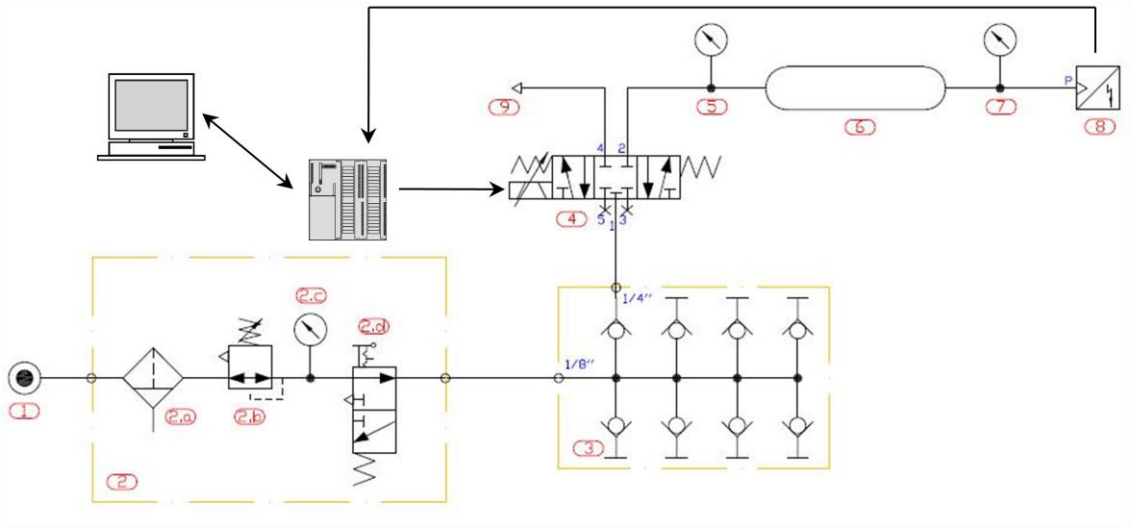

Figura 5.1. Sistema global.

En la **Figura 5.1** se pueden ver todos los elementos que engloba el sistema. El filtro regulador (ref. 2), es el encargado de suministrar 5 bares de presión en todo momento al sistema. El distribuidor (ref. 3), se encarga de suministrar diferentes canales neumáticos a la misma presión. La servoválvula (ref. 4), es la encargada de aumentar, disminuir o mantener la presión del depósito, en función del valor de presión que se introduzca en el PC. Luego están a lo manómetros (ref. 5 y 7), conectados a la entrada y salida del depósito (ref. 6), que sirven para contrastar los valores que da el sensor analógico de presión (ref. 8), conectado a la salida del depósito.

Como se puede observar en la **Figura 5.1**, es la señal del sensor de presión la que se realimentará a través del PLC (ref. 8). El sensor está conectado a la entrada analógica **PEW 754**. Este valor del sensor se puede ver en el PC en todo momento.

Cuando se desee cambiar el valor de presión se escribirá en el PC el **valor de consigna** que se desee. El PLC realizara las operaciones de control pertinentes con la **función FB 41 CONT_C** y a través de la salida analógica **PAW 754**, saldrá **la señal de control** que hará que la servoválvula aumente o disminuya la presión del depósito. Este cambio de presión es realimentado al PLC, de modo que cuando llegue al valor de consigna la servoválvula se detenga, para que la presión del depósito se mantenga.

La regulación descrita anteriormente no podría suceder sin la **función FB41 CONT_C**, integrada dentro del autómata. Esta función de control es de tipo PID, debido a sus tres acciones básicas de control.

5.3. Acciones básicas de control.

La misión de la función PID integrada en el autómata de SIEMENS consiste en comparar el valor de consigna, que introduce el operario a través del PC, con el valor real de la magnitud de salida de la planta, es decir, el valor de presión que mide el sensor. Esta comparación

genera la señal de control más adecuada para minimizar los errores y obtener una respuesta lo más rápida posible ante variaciones de consigna o ante perturbaciones exteriores.

La acción de control que deberá ejercer el controlador para conseguir las prestaciones antes mencionadas depende del tipo de planta a controlar. Sin embargo, los controladores suelen obedecer a unos pocos modelos básicos de comportamiento o combinaciones simples de ellos. Dichos comportamientos se denominan **acciones básicas de control**, y son las siguientes:

- Acción proporcional.
- Acción integral.
- Acción derivativa.

Estas acciones no suelen emplearse de forma aislada, sino combinadas entre sí para obtener las prestaciones deseadas.

5.3.1. Acción proporcional.

En un controlador de tipo proporcional puro [10] (**Figura 5.2**), la salida o acción de control $C(t)$ depende de la señal de error $\varepsilon(t)$, según la siguiente expresión:

$$C(t) = K_p \varepsilon(t) \tag{5.1}$$

Donde K_p es una constante, denominada **ganancia** o **constante proporcional.**

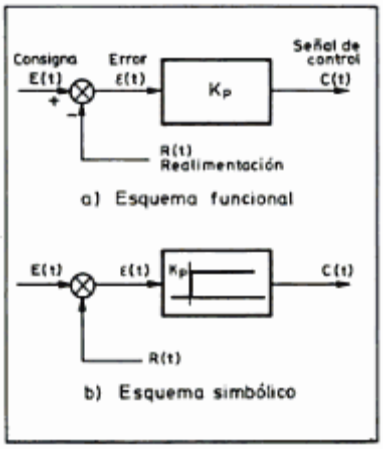

Figura 5.2. Bloque de control proporcional.

Hablando en términos de funciones de transferencia, expresadas por la relación de transformadas de Laplace, la expresión FDT para un bloque proporcional sería un constante:

$$T(s) = \frac{C(s)}{\varepsilon(s)} = K_p \tag{5.2}$$

Estos bloques proporcionales se denominan controladores de tipo **P** y se caracterizan por el hecho de que es necesaria la existencia de un error (ε) para que exista una acción de control. Según estas expresiones el error será más pequeño cuanto mayor sea la ganancia:

$$\varepsilon(s) = \frac{1}{K_p} C(s) \tag{5.3}$$

Pero hay que tener en cuenta el proceso a controlar, ya que la ganancia requerida para reducir el error en régimen permanente puede ser incompatible con la estabilidad relativa del sistema.

En ciertos procesos no hace falta tener en cuenta la estabilidad, ya que se puede trabajar con una ganancia elevada sin tener ningún tipo de problema. Pero en general hace falta combinar varias acciones para conseguir una respuesta óptima.

5.3.2. Acción integral.

El principal inconveniente de un regulador que tuviera sólo la acción P es que deja siempre un error por corregir. La acción integral permite anular este error, haciendo que la señal de control, $C(t)$, crezca proporcionalmente al producto (error $\times$ tiempo). Se puede decir, pues, que un sistema con acción integral tiende a anular el error promedio, en otras palabras mejorará el régimen permanente del sistema.

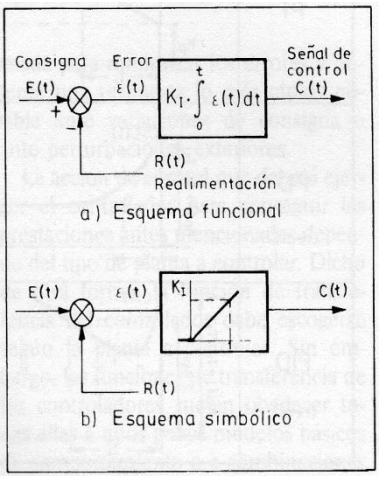

Figura 5.3. Bloque de control integral.

El comportamiento de la estructura de bloques de la **Figura 5.3** [10] puede expresarse analíticamente con una de las siguientes ecuaciones:

$$d\left[C(t)\right] = K_I \varepsilon(t) dt \tag{5.4}$$

$$C(t) = K_I \int_0^t \varepsilon(t)dt \tag{5.5}$$

La ecuación (5.5) indica que, en tanto que exista error entre la **consigna** y la **realimentación**, el sistema aumenta la salida, intentando la corrección del mismo. En el momento en que el error es nulo, el sistema mantiene el valor de salida constante.

En términos de función de transferencia, aplicando la transformación de Laplace a cualquiera de las ecuaciones (5.3) o (5.4) resulta:

$$T(s) = \frac{C(s)}{\varepsilon(s)} = \frac{K_I}{s} = \frac{1}{T_I s} \tag{5.6}$$

La constante K_I se denomina **constante integral** y su inversa T_I se denomina **constante de tiempo integral**.

Los controladores tipo **I** tienden, como se ha dicho anteriormente, a eliminar el error pero en ciertos procesos pueden causar inestabilidad o una respuesta dinámica muy lenta, en los cuales, es necesario otra acción de control.

5.3.3. Acción derivativa.

Las acciones proporcional e integral vistas anteriormente no permiten resolver de forma satisfactoria todos los problemas de control. De la **acción P** se puede decir que siempre deja un error permanente y de la **acción I** que puede causar inestabilidad o exceso de tiempo de respuesta. Es por esto que existe la **acción D**, para complementarlas, ayuda a obtener una respuesta dinámica más rápida (tiempo de respuesta menor), en otras palabras, mejora el régimen transitorio del sistema.

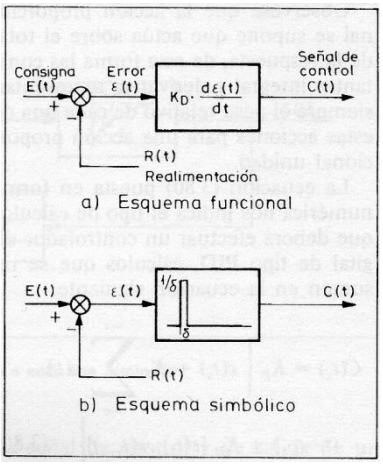

Figura 5.4. Bloque de control derivativo.

La **Figura 5.4** [10], representa la estructura de bloques de un regulador de acción derivativa. Dicha acción derivativa se caracteriza por generar una señal de control proporcional

a la tasa de variación del error con el tiempo. Esto se puede expresar analíticamente con la siguiente ecuación:

$$C(t) = K_D \frac{d\varepsilon(t)}{dt} = T_D \frac{d\varepsilon(t)}{dt} \tag{5.7}$$

Donde K_D se denomina **constante de acción derivativa** y es igual a las **constante de tiempo derivativa** T_D

En términos de función de transferencia, aplicando la transformación de Laplace a la ecuación (5.6) resulta:

$$T(s) = \frac{C(s)}{\varepsilon(s)} = T_D s = K_D s \tag{5.8}$$

Los circuitos derivadores reales suelen introducir un cierto retardo ya que el valor de $d\varepsilon/dt$ en el instante t=0 debería ser un impulso de amplitud infinita y duración cero (impulso unitario o de Dirac). Es de prever que este tipo de respuesta, con valor infinito, será imposible de conseguir con componentes reales.

5.4. Controladores PID.

El control automático sienta sus bases esencialmente en el concepto de realimentación [2]. El error de regulación es consecuencia de esta realimentación y, además, es la base a partir de la cual actúa el **PID** y por tanto, se intuye que cuanto más precisa sea la medida, mejor se podrá controlar la variable en cuestión. Esta es la razón por la que el sensor es el elemento crítico del sistema. También se debe tener en cuenta la instalación, especialmente en la forma en que se transmiten los datos del sensor hacia el regulador y posibles fuentes de interferencias.

Un regulador **proporcional-integral-derivativo** o **PID** tiene en cuenta el error, la integral del error y la derivada del error, es decir, se utilizan las tres acciones antes mencionadas. La acción de control se calcula multiplicando los tres valores (K_p, K_i, K_d) por una constante y sumando los resultados, como se puede observar en la **Figura 5.5**. Los valores de las constantes (vistas en la sección anterior) definen el comportamiento del regulador.

En otras palabras un controlador PID, corresponde a la superposición de las tres acciones básicas de control. El comportamiento desde el punto de vista temporal, según la **Figura 5.5**, sería, el siguiente:

$$C(t) = K_p \left\{ \varepsilon(t) + \frac{1}{T_I} \int_0^t \varepsilon(t)dt + T_D \frac{d\varepsilon(t)}{dt} \right\} \tag{5.9}$$

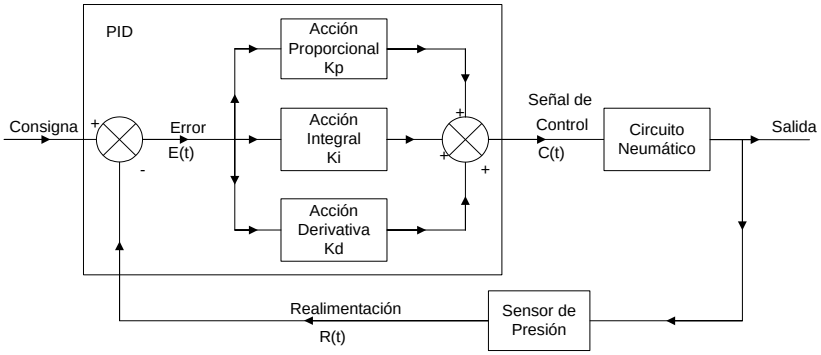

Figura 5.5. Diagrama de bloques del sistema.

De la ecuación anterior podemos deducir la siguiente función de transferencia:

$$T(s) = K_P \left\{ 1 + \frac{1}{T_i s} + T_D s \right\} = K_P \left\{ 1 + \frac{K_I}{s} + K_D s \right\} \qquad (5.10)$$

Se puede observar que la acción proporcional actúa sobre el total de la respuesta, de esta forma las constantes integral y derivativa representan siempre el peso relativo de cada una de estas acciones para una acción proporcional unidad

En la mayor parte de casos la aplicación de controladores industriales se resolverá mediante un hardware estándar y el usuario deberá únicamente programar o ajustar las constantes K_P, K_I y K_D. Sin embargo la elección de dichos parámetros no es simple, pues suelen aparecer problemas de inestabilidad. Pero gracias a las reglas de Ziegler-Nichols se pueden hallar estos valores sin mucha dificultad.

5.5. Regulación PID en la estructura SIEMENS.

Cuando el proceso a controlar sea simple y lento (Presión, nivel, caudal, temperatura), el control se realiza con las funciones PID que están integradas en el PLC, en este caso la función que se utilizara es la **FB 41 CONT_C** [11], que sirve para la regulación de procesos industriales con magnitudes de entrada y salida continuas.

Por otra parte, hay casos en los que los lazos son más complejos y es necesario software complementario al STEP7, incluso, si la complejidad lo requiere, se pueden diseñar tarjetas especiales para dichos procesos. Estos serian los casos de las gamas media y alta de regulación, respectivamente.

En el caso de este proyecto, cuya regulación es básica, la función **FB 41 CONT_C** puede regular fácilmente procesos como el nivel, la presión o el caudal.

Esta función se puede utilizar de muchas maneras en un proceso de regulación, como regulador con consigna fija o formando parte de una estructura de lazos complejos (cascada, mezcla, etc.). En este caso se utilizará en consigna fija.

Este bloque de funcion se basa en un algoritmo matemático PID, por tanto tendrá que ejecutarse cíclicamente, y los resultados obtenidos de dicho algoritmo, así como los valores de sus parámetros, se guardaran en una DB. Por lo tanto, por cada función FB41 necesitaremos una DB de instancia.

5.5.1. Regulación continua con la función FB 41“CONT_C”.

Cuando se proceda a programar la **DB** de instancia de esta función en la aplicación STEP7, mostrará el aspecto de la **Figura 5.6**:

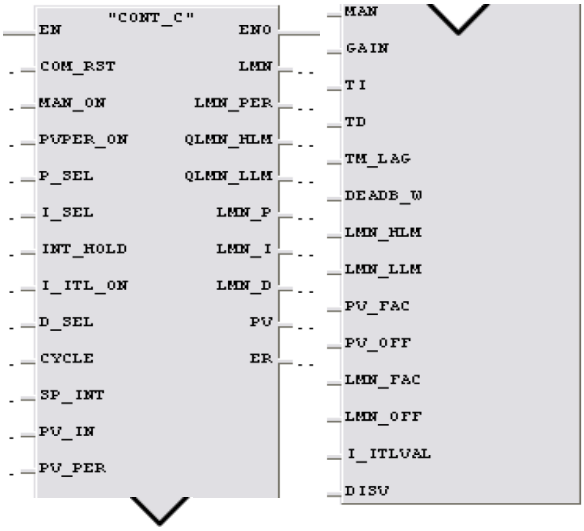

Figura 5.6. Aspecto de la función FB 41“CONT_C".

Tal y como se puede apreciar en el bloque de la **Figura 5.6**, el lenguaje utilizado en la aplicación STEP7 para realizar el control del proceso es el KOP. Se está trabajando con señales analógicas pero sin realizar grandes trozos de códigos, además gráficamente es más fácil de comprender este bloque en KOP que en los otros dos tipos de lenguajes que permite el STEP7.

En este bloque, los parámetros de la izquierda son las **ENTRADAS** de la función; algunos de los cuales son solo de lectura, otros de lectura y escritura, otros son los parámetros PID, etc.

En cambio, todos los parámetros situados a la derecha del bloque, son las **SALIDAS,** y todas son de lectura. La función asigna como valores de salida toda la información de la regulación que se está realizando.

5.5.2. Esquema de bloques de la función PID FB41 "CONT_C".

Para poder entender la función **FB 41**, hay que mirar el esquema de bloques de la **Figura 5.7** [11]. Este esquema se va a dividir en tres partes; una parte superior, una intermedia y una inferior.

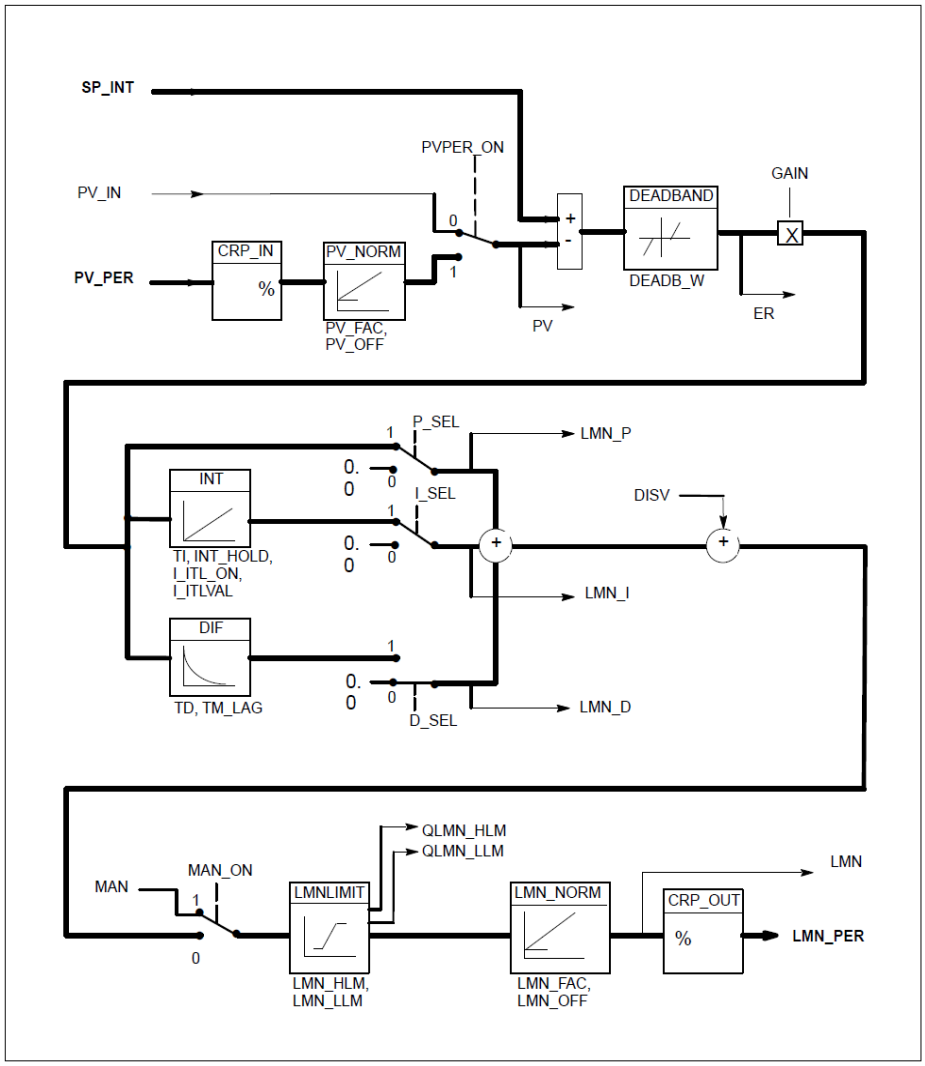

Figura 5.7. Esquema de bloques.

- **Parte superior**: En esta parte se realiza la comparación entre la consigna y el valor real del proceso normalizado, y así crear el error de regulación.

- **Parte intermedia**: Se pueden conectar o desconectar las funciones parciales del regulador (P, PI y PID), que se aplicaran al error originado en la parte superior. Se introducen los valores de los parámetros de cada una de las acciones (K_P , K_I y K_D) y se lee la salida que corresponde a cada parámetro y, teniendo en cuenta la perturbación DISV, originar la salida real del regulador.

- **Parte inferior**: En esta última parte se obtiene la salida que origina el regulador, ya sea manual o automáticamente. Se pueden establecer límites para esta señal de control y se transforma en números reales y en formato de periferia.

5.5.2.1. Descripción de los parámetros de la parte superior.

En la **Figura 5.8** se puede observar la entrada de esta función, en la que se genera el error de regulación.

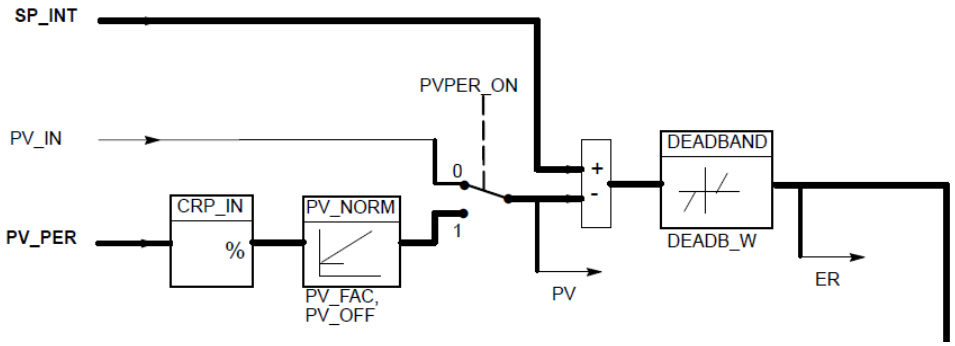

Figura 5.8. Parte superior de la Función FB41 "CONT_C".

SP_INT: con este parámetro se fija la consigna del lazo de regulación; se tratará de un número real y, normalmente, de un porcentaje, de -100.0 % hasta el 100.0 %(-10v a 10v) si es bipolar y de 0 a 100% (0 a 10v) si es unipolar. Existe la opción de introducir directamente el valor de la magnitud física, pero en este caso habría que adaptar los parámetros **PV_FAC** y **PV_OFF** de la función **PV_NORM**, tal y como se verá más adelante. El valor inicial de este parámetro es cero, y es de lectura-escritura.

PVPER_ON: mediante este parámetro binario definiremos el modo de lectura al valor del proceso (PV). Si es "0", leeremos el valor del **PV** desde el parámetro **PV_IN**; si es "1", se leerá desde el parámetro **PV_PER**, como de costumbre. Es un valor de lectura – escritura, y el inicial es "0".

PV_PER: en este parámetro recibiremos la información que nos proporcione directamente un sensor, es decir, el valor de la lectura de una tarjeta analógica. Así, este parámetro guardará la dirección de la entrada a la que está conectada el sensor (PEW…). Cuando el sensor sea de una única polaridad (0…10 V) y la medición se encuentre en rango nominal, el valor estará comprendido entre 0 y 27648 y, cuando sea bipolar, el valor se encontrará entre -27648 y 27648 (-10v…10v).

CPR_IN: esta función transforma los valores enteros de **PV_PER** a valores reales, comprendidos entre -100% y 100%, para el caso bipolar y entre 0 y 100%, para el caso unipolar. Para realizar este cambio se aplica la siguiente fórmula:

$$CPR\_IN = PV\_PER\frac{100}{27648} \tag{5.11}$$

PV_NORM: esta función normaliza los porcentajes de los valores proporcionados por el sensor, adecuando los parámetros PV_FAC y PV_OFF, tal y como se puede apreciar en la siguiente fórmula:

$$PV\_NORM(Salida) = CPR\_IN(Salida) \times PV\_FAC + PV\_OFF \quad (5.12)$$

PV_FAC: factor del valor del proceso. Multiplica la salida de la función CPR_IN, para adaptar los valores proporcionados por el sensor a un nuevo rango. Su valor puede ser cualquier número real, pero inicialmente será 1.

PV_OFF: el offset del valor del proceso. Añade un valor a la salida de la función CPR_IN para adaptarlo al nuevo rango. Su valor puede ser cualquier número real, pero normalmente será 0.

PV_IN: en cuanto a esta entrada, tendríamos que adaptar el valor entero que nos proporciona el sensor con las demás funciones del autómata para convertirlo en valor real, y para que, de este modo, la función realice una correcta comparación con SP_INT. Se utiliza cuando el valor del PV es real (entre el -100.0 % y el 100.0 %) o haya sido adaptado al valor de la magnitud física (en formato real DWORD).

PV: la lectura del valor del proceso se guarda en este parámetro y estará en números reales, porcentajes (-100.0 % - 100.0 %) o en cualquier otra unidad. Es sólo de lectura.

DEADB_W: después de realizar la comparación, podremos aplicar el DEAD_BAND a lo que vaya a ser el error. Por ejemplo, si se quiere eliminar las fluctuaciones o el "ruido" que puede provocar un sensor. Así pues, introduciendo el número real en este parámetro, definiremos la amplitud de esta banda. El valor inicial es 0.0 y eso significa que DEAD_BAND está desconectado.

Cuanto más ancha sea esta banda, menos precisión tendrá la regulación, ya que, mientras el error no salga de esta banda, el regulador no se dará cuenta de que ha habido alguna variación en el error.

En la **Figura 5.9** se puede ver como se relacionan los parámetros de creación de error (ER), consigna, PV y DEAD_BAND.

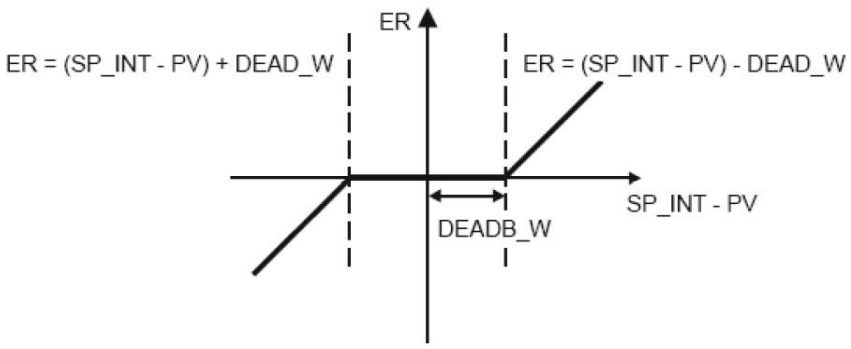

Figura 5.9. Error con Dead Band.

Cuanto más ancha sea esta banda, menos precisión tendrá la regulación, ya que, mientras el error no salga de esta banda, el regulador no se dará cuenta de que ha habido alguna variación en el error.

ER: error de regulación. Es sólo de lectura y hace referencia al error de lazo del proceso que estemos controlando en el momento. Su valor estará en números reales.

5.5.2.2. Descripción de los parámetros de la parte intermedia.

Se puede observar en la parte del diagrama de bloques que corresponde a los parámetros de configuración del algoritmo de regulación PID. Los parámetros **D** e **I** pueden conectarse y desconectarse individualmente, mientras que la acción **P** estará activada mientras haya señal de entrada.

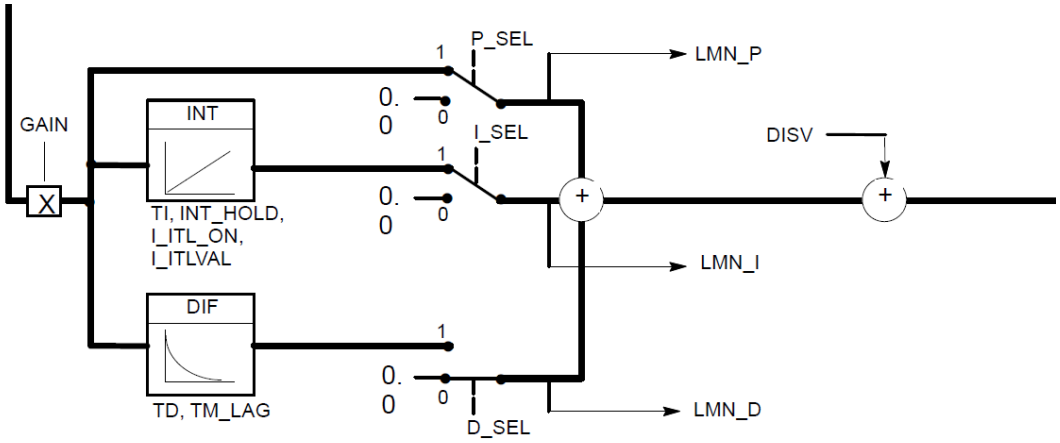

Figura 5.10. Parte intermedia de la función FB41 "CONT_C".

El funcionamiento se basa en el algoritmo matemático de la expresión 5.9.

GAIN: la ganancia proporcional del regulador será (K_P), y puede adquirir cualquier valor real positivo o negativo.

La acción proporcional puede ser directa o inversa, es por esto que este parámetro tiene la opción de incluir números negativos o positivos. En el caso de este proyecto existe relación directa entre la consigna y la variable a controlar, ya que si aumentamos la consigna aumenta también la presión del depósito y viceversa.

Si el valor de la ganancia es cero el valor de salida será también cero, independientemente de los valores T_i y T_d.

Función INT: este bloque da la salida del regulador correspondiente al efecto integral, según los parámetros **Ti**, **IN_HOLD**, **I_ITL_ON**, **I_ITLVAL**.

Ti: es el tiempo de la acción integral y su unidad se fijará en segundos. El valor mínimo de este parámetro está relacionado con el parámetro **CYCLE** (**Ti** ≥ **CYCLE**) y tendrá un valor inicial de 20 segundos (T#20S).

INT_HOLD: cuando este parámetro binario se activa, se congela la salida del integrador en ese momento, independientemente de la evolución del error.

I_ITLVAL: La salida del integrador puede forzarse en cualquier momento, mediante este parámetro de número real. El valor inicial es 0.0. Este parámetro está relacionado con el parámetro **I_ITL_ON**.

I_ITL_ON: Cuando este parámetro binario este activado se forzara la salida del integrador introducida en el parámetro **I_ITLVAL.**

Función DIF: este bloque corresponde al efecto derivativo y se puede configurar según los parámetros **Td** y **TM_LAG**.

Td: es el tiempo de la acción derivada y su unidad se fija en segundos. El valor mínimo de este parámetro está relacionado con el parámetro **CYCLE** (**Td** ≥ **CYCLE**) y tendrá un valor inicial de 10 segundos (T#10S).

TM_LAG: el efecto de la acción derivativa se puede retrasar con este parámetro, según el tiempo introducido en el mismo. El valor mínimo de este parámetro está relacionado con el parámetro **CYCLE** (**TM_LAG** ≥ **CYCLE/2**) y tendrá un retardo inicial de 2 segundos (T#2S).

P_SEL, **I_SEL** y **D_SEL**: mediante estos parámetros binarios, se pueden configurar los diferentes tipos de reguladores (P, I, PI, PD o PID). El regulador predeterminado en la configuración inicial es del tipo PI, por lo tanto, **P_SEL** = "1**", I_SEL** = "1" y **D_SEL** = "0".

LMN_P, **LMN_I** y **LMN_D**: estos parámetros reales son sólo de lectura y son las salidas de las tres acciones del regulador. Habrá señal de salida mientras que este activada la acción que corresponde de cada uno de estos tres parámetros.

DISV: se podrá simular una perturbación en el proceso con este parámetro, sumando o restando un valor a la salida que origina el regulador.

5.5.2.3. Descripción de los parámetros de la parte inferior.

En el diagrama de bloques de la **Figura 5.11** aparecen los parámetros de configuración de la salida originada por el regulador, se puede utilizar el modo manual o automático, establecer límites y modificar la salida provisional en porcentajes, números reales y en formato de periferia.

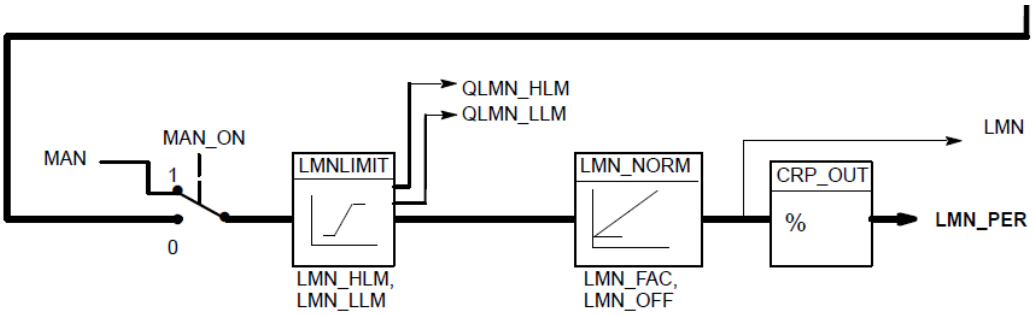

Figura 5.11. Parte inferior de la función FB41 "CONT_C".

MAN_ON: este regulador, al igual que todos, tiene la opción de funcionar en modo manual o en modo automático. Si este bit fuese un "1", el valor escrito en el parámetro **MAN** se transferiría a la salida del regulador. Por ejemplo, si escribimos en el parámetro MAN el número 50, como estos números son porcentajes de la salida unipolar (0-10v), la salida dará 5v.

Cuando el bit **MAN_ON** es "0" es cuando se ejecuta el algoritmo PID de la **Figura 5.10**.

Función LMNLIMIT: en el caso de que sea necesario establecer límites a la salida del regulador, se podría hacer mediante los parámetros **LMN_HLM y LMN_LLM**.

LMN_HLM: es el límite superior que generará el regulador. Funcionando tanto manual como automáticamente, este límite no se podrá superar nunca. Cuando el valor de salida alcanza el valor de **LMN_HLM**, se activa el parámetro binario **QLMN_HLM**, lo cual indica que se ha llegado al valor máximo de salida.

LMN_LLM: es el límite inferior que generará el regulador. Funcionando tanto manual como automáticamente, el límite de salida nunca será menor que el parámetro **LMN_LLM**. Cuando el valor de salida alcanza el valor de **LMN_LLM**, se activa otro parámetro binario, **QLMN_LLM**, lo cual indica que hemos llegado al valor mínimo de salida.

Función LMN_NORM: se utiliza para normalizar la salida originada por el regulador y convertirlo en porcentajes, según la función de entrada **PV_NORM**, conforme a la siguiente fórmula:

$$LMN = LMNLIMIT \times LMN\_FAC + LMN\_OFF \qquad (5.13)$$

LMN_FAC: con este parámetro se puede modificar la salida del regulador. Es un número real.

LMN_OFF: con este parámetro se puede añadir un offset a la señal de salida. Al igual que el anterior parámetro es un número real.

LMN: es la salida de la función **LMN_NORM**, y proporciona la salida en porcentajes (sólo de lectura).

Función CPR_OUT: convierte automáticamente el valor real de la salida en un número entero y este resultado lo asigna al parámetro **LMN_PER**, aplicando la siguiente fórmula:

$$LMN\_PER = LMN\,\frac{27648}{100} \tag{5.13}$$

5.5.2.4. Otros parámetros de la función FB41.

Aunque se ha detallado el diagrama de bloque de la **Figura 5.7,** quedan dos parámetros que merece la pena destacar. Estos parámetros aparecen en la **Figura 5.6**y son **CYCLE** y **COMRST** [11].

CYCLE: con este parámetro se puede dar un valor a la frecuencia de muestreo, es decir, la frecuencia con la que se llama al bloque **FB41.**

El cálculo de los valores en este bloque de regulación sólo será correcto si el bloque se llama en intervalos regulares. Por esta razón se debería llamar a los bloques de regulación desde una OB de alarma cíclica (OB30 a OB35). El intervalo se deberá especificar en el parámetro **CYCLE**.

COMRST: con este parámetro binario se resetea o reinicia el bloque **FB41**, cuando se le da el valor de 1. Este reset se puede realizar en cualquier momento y lo más adecuado sería programar este reset en el OB100, que es de rearranque. Al arrancar la CPU (p.ej. tras conmutar el selector de modo de operación de STOP a RUN o al conectar la tensión de red) el bloque de organización OB100 se procesa antes de la ejecución cíclica del programa (OB1).

Después de activar este parámetro, los valores asignados a **LMN_P**, **LMN_I** y **LMN_D** se borran y, por tanto, la salida también es cero. Si se mantiene activado este parámetro, el regulador se bloquea, por lo que resulta necesario que durante la regulación sea “0”.

5.6. Configuración del Hardware del PLC.

Una vez se ha explicado el funcionamiento del Bloque PID, se procederá a configurar y a programar el PLC mediante la aplicación SIMATIC Manager de la siguiente forma:

- En esta sección se elegirá de la biblioteca interna de SIMATIC el hardware utilizado en el laboratorio, es decir, el PLC S7-300 con CPU compacta 314C-2DP.
- En la **sección 5.7**. se configurarán las E/S analógicas.

- En la **sección 5.8**. se programará la alarma cíclica OB35 y el reset del regulador en el OB100.
- Se creará una tabla de variables en la **sección 5.9.**
- Por último se configurará el entorno grafico "**Parametrizar Regulación PID**" en la **sección 5.10.**

En primer lugar se ejecuta el entorno de programación SIMATIC Manager y aparecerá la **Figura 5.12**, en la cual sale el asistente de STEP 7. Se le dará a cancelar para no utilizarlo.

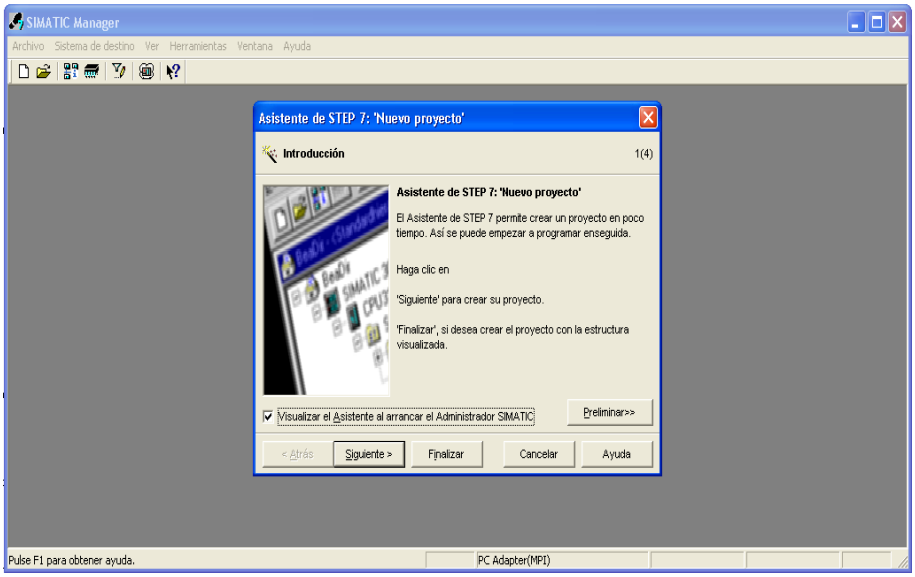

Figura 5.12. SIMATIC Manager.

Se creara un nuevo documento haciendo click en la pestaña marcada de la **Figura 5.13**, donde aparecerá una ventana en la cual se le podrá poner el nombre y la ruta.

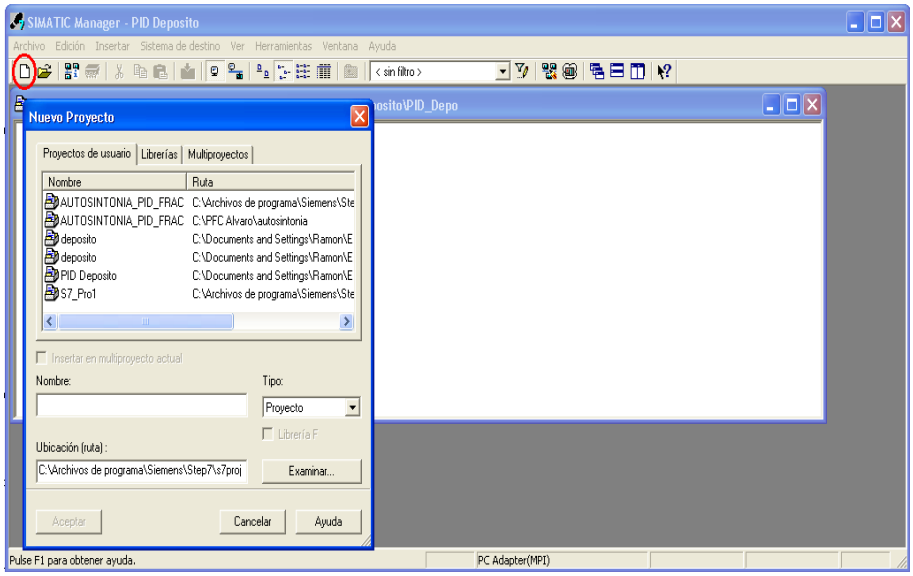

Figura 5.13. Nuevo documento.

Se ha optado por nombrar al documento como PID Deposito, como se puede ver en **Figura 5.14:**

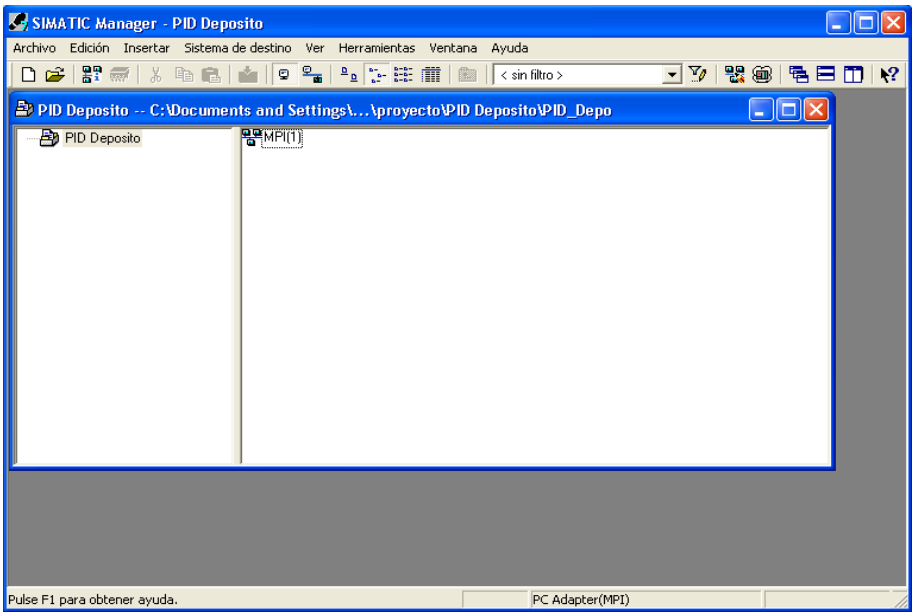

Figura 5.14. Ventana SIMATIC.

Una vez se le ha dado el nombre al documento, se introduce el autómata que se ha utilizado en el proyecto, es decir, un PLC de la gama S7-300. Como se ve en la **Figura 5.15:**

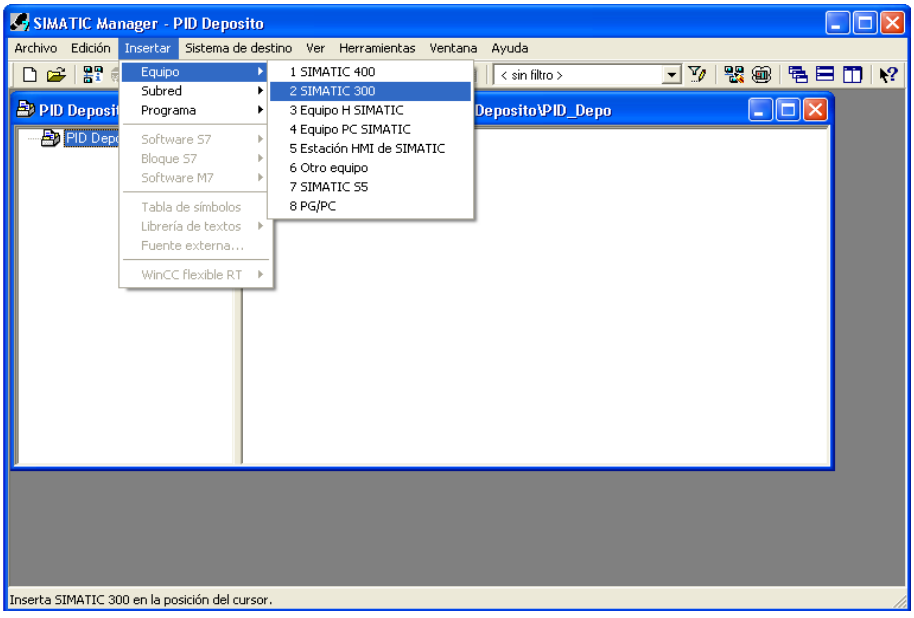

Figura 5.15. SIMATIC S7-300.

Una vez se ha hecho esto, la ventana mostrará que se ha introducido el elemento SIMATIC S7-300. Como se puede ver en la **Figura 5.16:**

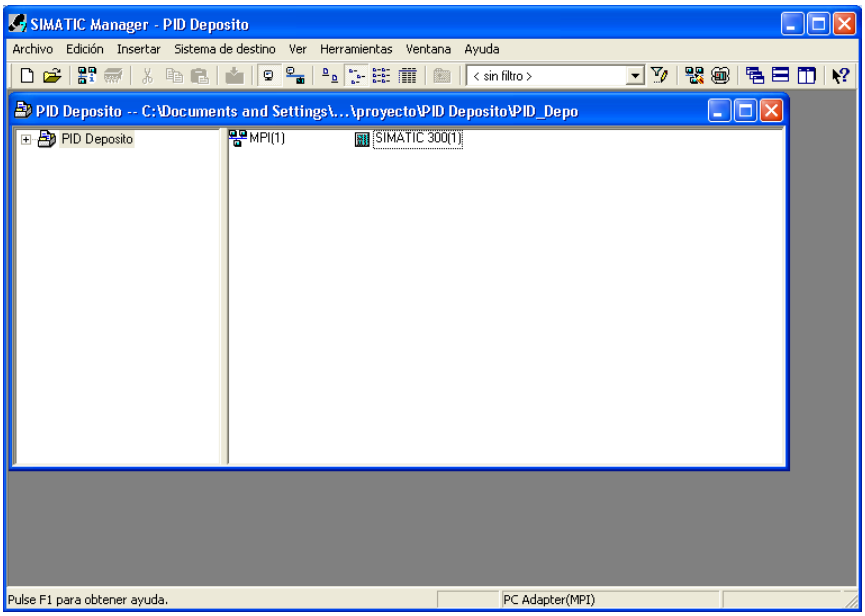

Figura 5.16. Elemento SIMATIC 300.

Una vez se tiene el PLC de la gama que se ha utilizado, se tendrá que elegir en la biblioteca interna, el hardware del laboratorio que se ha utilizado. Se abre el elemento SIMATIC 300 haciendo doble click en él y se puede ver el elemento Hardware, como se ve en la **Figura 5.17:**

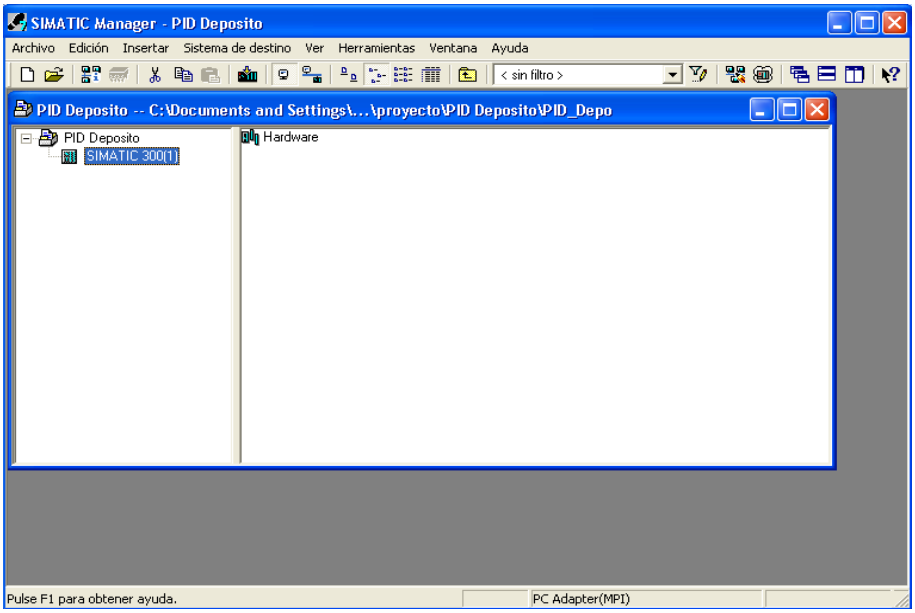

Figura 5.17. Hardware.

Una vez ha aparecido este elemento, se vuelve hacer doble click en él y aparecerá otra ventana en la que se podrá configurar el PLC (**Figura 5.18**).

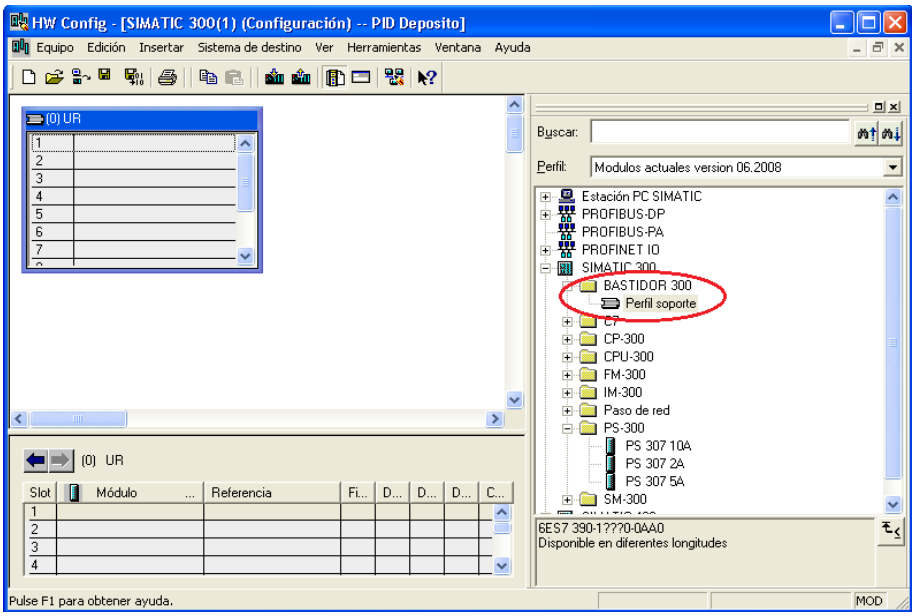

Figura 5.18. Configuración del PLC.

Una vez en esta ventana se tiene que introducir el **perfil de soporte** que se encuentra resaltado en la **Figura 5.18**, para poder introducir la CPU compacta 314C-2DP con periferia integrada, cuyo número de serie es 6ES7314-6CG03-0AB0. Esta CPU viene resaltada en la **Figura 5.19.**

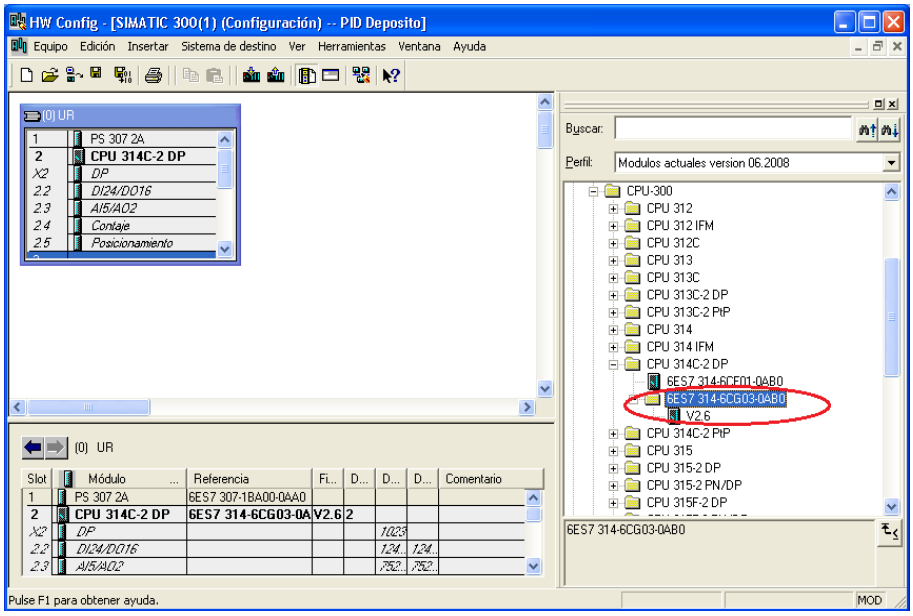

Figura 5.19. CPU.

La función PID FB41 será programada dentro de una OB periódica, ya que estos bloques se pueden programar para que se ejecuten a intervalos regulares de tiempo. En la gama S7300 la única que se encuentra habilitada para esta función es la OB35. Por tanto, se entra en las propiedades de la CPU, como muestra la **Figura 5.20**.

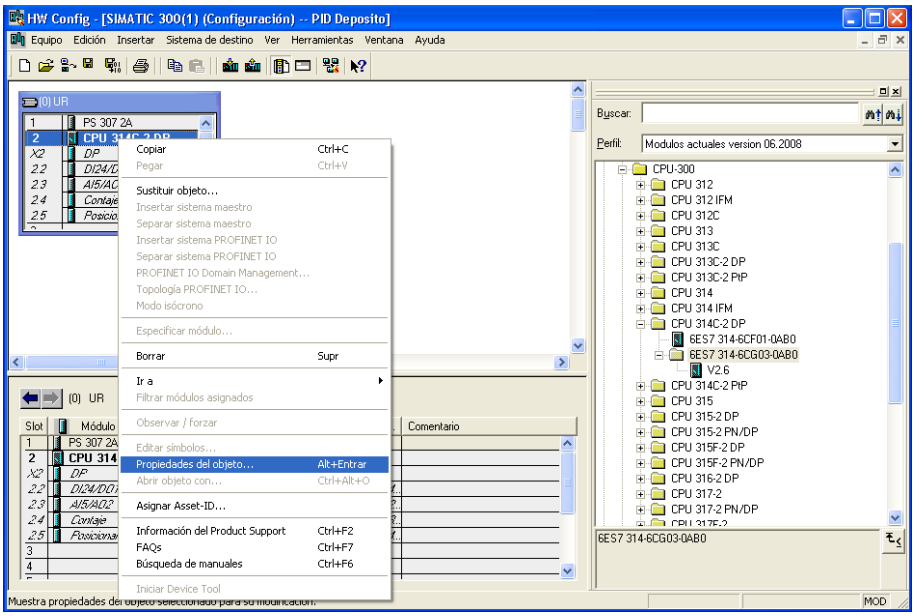

Figura 5.20. Acceso a propiedades de la CPU.

La OB35 se suele utilizar para las llamadas a lecturas analógicas, regulación PID, y otros procesos que requieran un tratamiento uniforme en el tiempo, e independiente del tiempo de ciclo del programa.

Como se puede ver en la **Figura 5.21** aparece la ventana de propiedades de la CPU. En esta ventana se selecciona la pestaña de alarmas cíclicas, en donde se ve como la OB35 tiene una propiedad de 12 y un periodo de ejecución (Periodicidad) de 100ms, que habrá que cambiar en función de la duración del ciclo del autómata y del sistema a controlar. En el caso de este proyecto se ha determinado que el tiempo sea de 20ms [12].

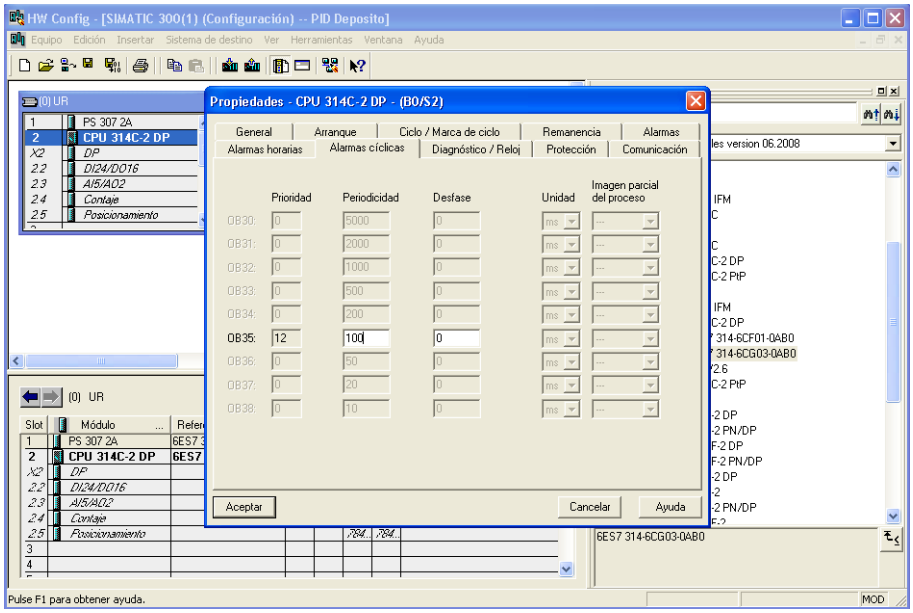

Figura 5.21. Propiedades de la CPU.

5.7. Configuración de las E/S analógicas.

Las tarjetas analógicas del S7 soportan diferentes tipos de medidas, configurándose desde **Hardware** de SIMATIC Manager. Estas tarjetas pueden medir tensión, intensidad, temperatura y resistencia, y darán como salida señales de tensión o intensidad.

Para poder interpretar los valores del proceso, la señal que recibiremos del sensor y la que nos leerá la tarjeta deben de coincidir. El sensor analógico de presión da dos tipos de señales de salida, una de intensidad y otra de tensión. En este caso se ha utilizado la señal de tensión del sensor, por tanto, la entrada analógica tiene que poder leer esta señal.

En primer lugar se abrirá las propiedades de las E/S analógicas. Se puede ver este elemento resaltado en la **Figura 5.22:**

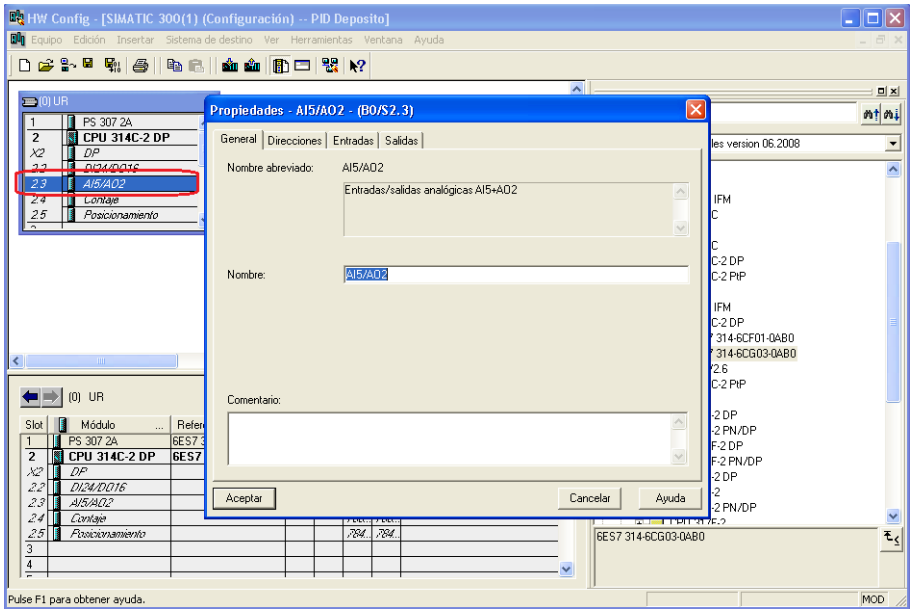

Figura 5.22. E/S analógicas.

En la pestaña de **direcciones** de la ventana de propiedades (**Figura 5.23**) se puede observar el direccionamiento de las E/S analógicas. Las entradas empiezan en PEW752 y terminan en la PEW761, mientras que las salidas empiezan en PAW752 y acaban en la PAW755. También se puede observar como estas direcciones son configurables.

En la misma ventana, en la pestaña de **entradas**, se encuentran las 5 entradas analógicas (**Figura 5.24**). Las 4 primeras se pueden configurar para medir tensión (**Figura 5.24**) o intensidad (**Figura 5.25**), cada uno con un rango especifico. En la última se puede conectar una termoresistencia u otro tipo de resistencia para controlar la temperatura (**Figura 5.26**). También existe la opción de que estén desactivadas para que la CPU no las lea y de este modo que el ciclo sea más rápido.

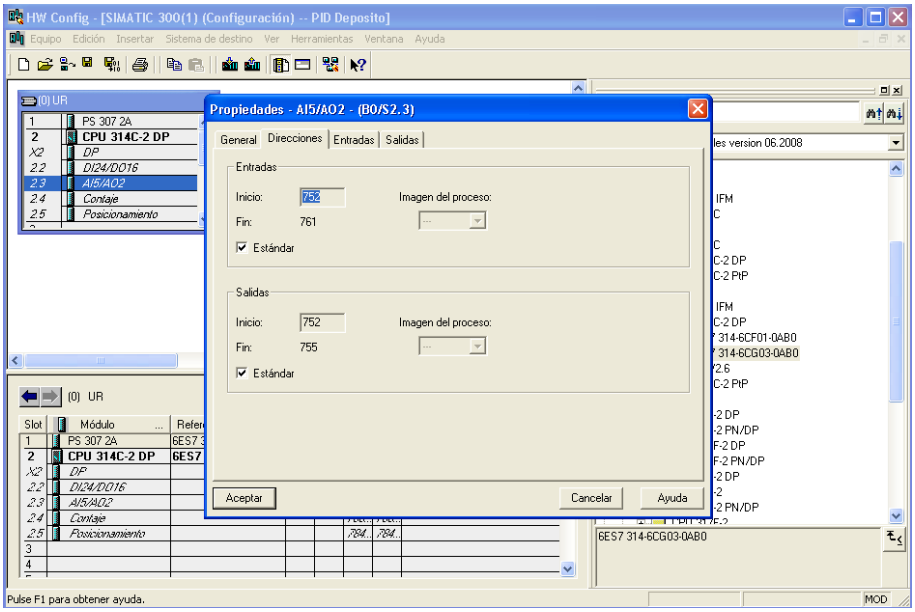

Figura 5.23. Direcciones de las E/S analógicas.

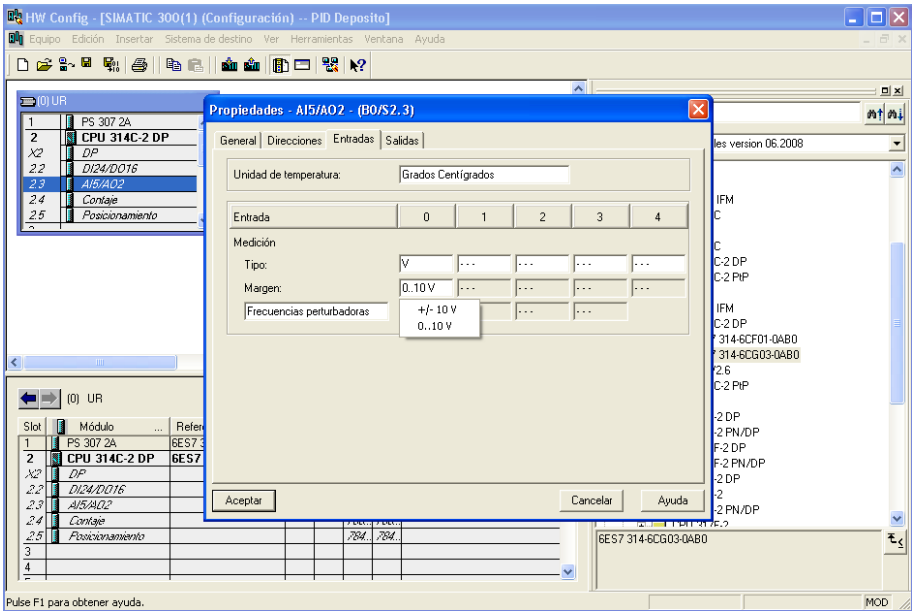

Figura 5.24. Lectura de tensión.

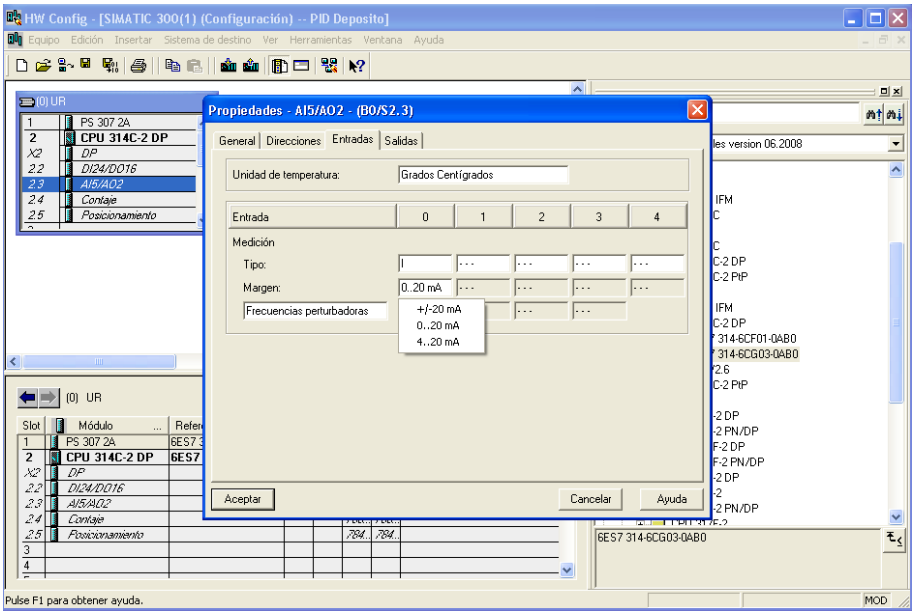

Figura 5.25. Lectura de corriente.

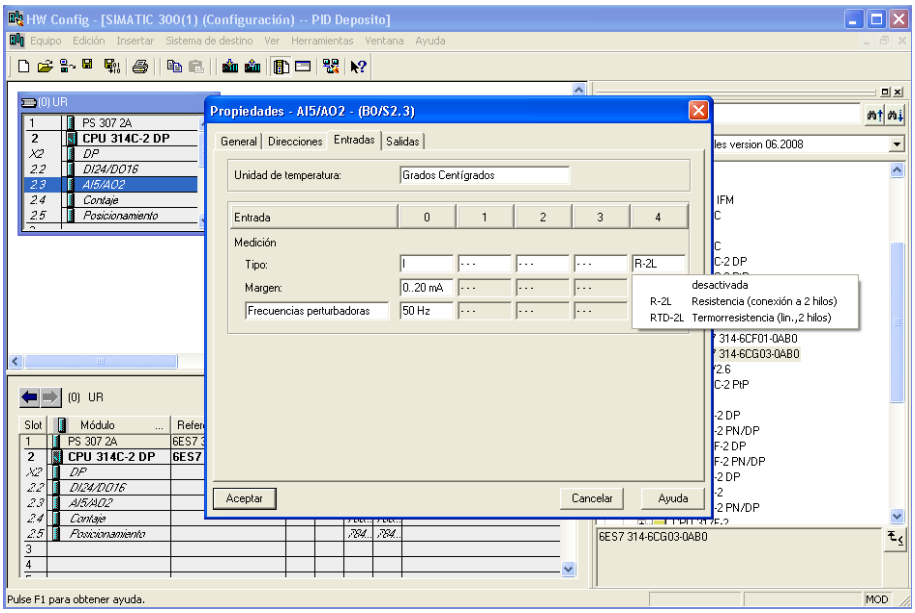

Figura 5.26. Lectura de temperatura.

Existe la posibilidad de que en la medición de las señales se puedan eliminar ruidos que distorsionen la señal (**Figura 5.27**).

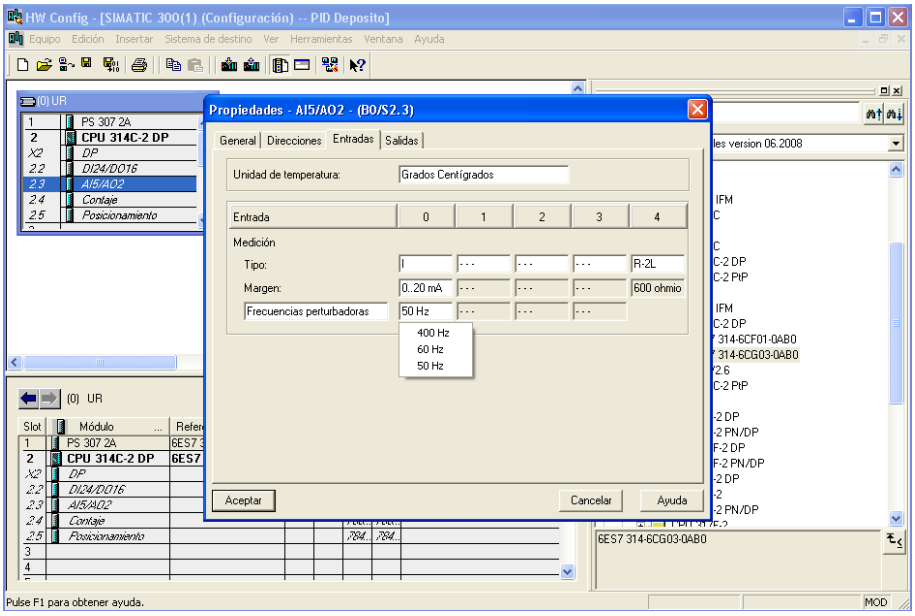

Figura 5.27. Frecuencias perturbadoras.

Como se ha mencionado antes, se desactivaran las entradas analógicas que no se utilicen, dejando solamente activada la primera entrada, midiendo esta tensión y de forma unipolar (**Figura 5.28**).

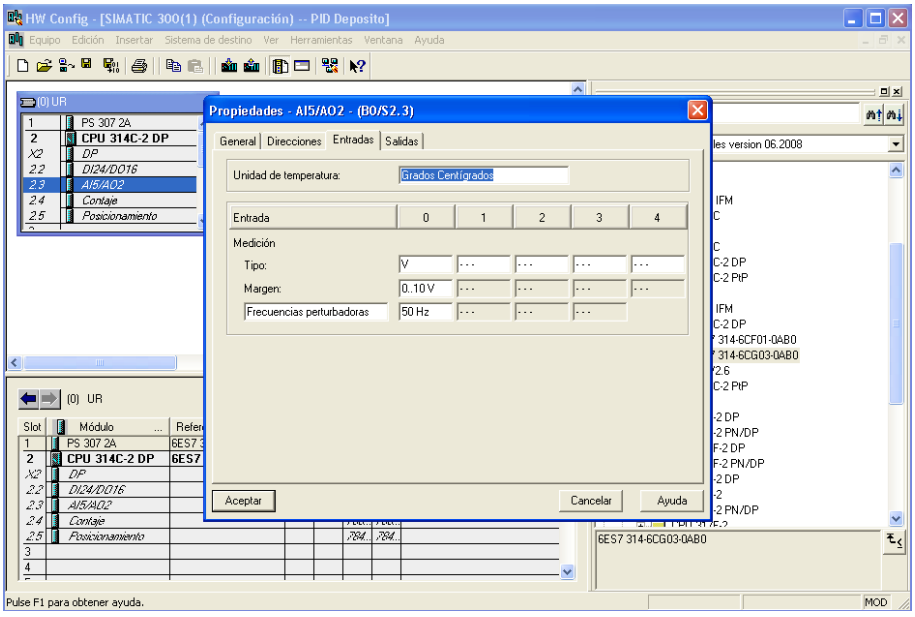

Figura 5.28. Entradas analógicas configuradas.

Si se cambia a la pestaña de **salidas**, estarán las dos salidas analógicas cuyas señales serán de tensión o de corriente, con el mismo rango de señal que el de las entradas. Al igual que con las **entradas**, se desactivará la salida que no se vaya a utilizar (**Figura 5.29**).

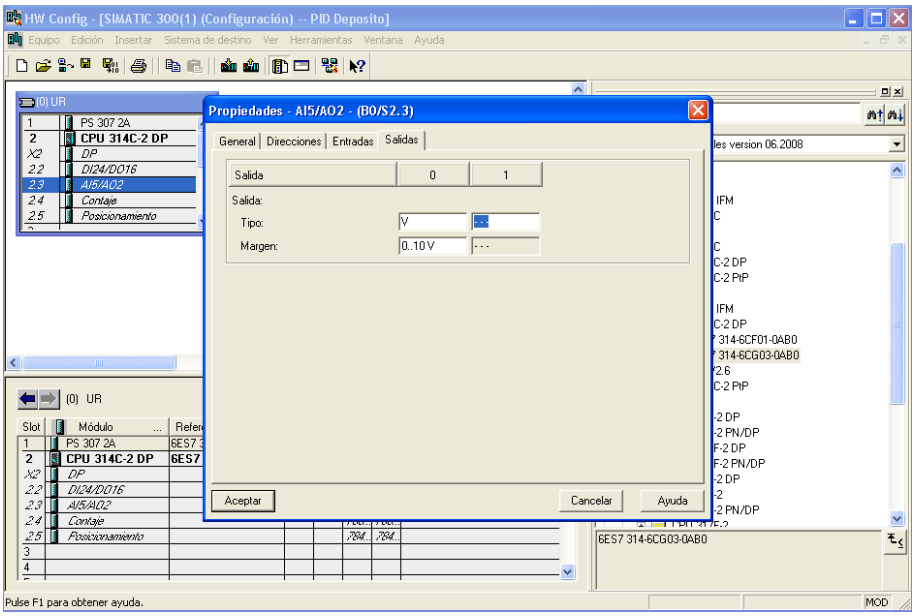

Figura 5.29. Salidas configuradas.

Una vez realizadas las pertinentes configuraciones, hay que compilarlas en la CPU, estando esta en modo **STOP**. Hay que hacer click en el botón de **cargar en el modulo** (**Figura 5.30)** y todas las anteriores configuraciones quedarán guardadas en la CPU.

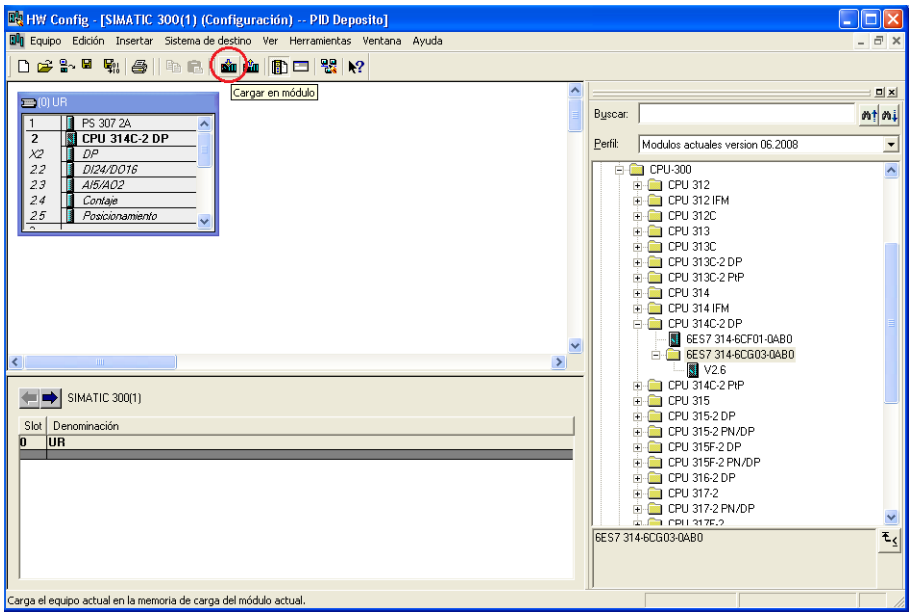

Figura 5.30. Compilación de las configuraciones.

En el caso de que la CPU este en modo **RUN**, como en la **Figura 5.31**, saldrá un mensaje en el que habrá que dar a **SI**.

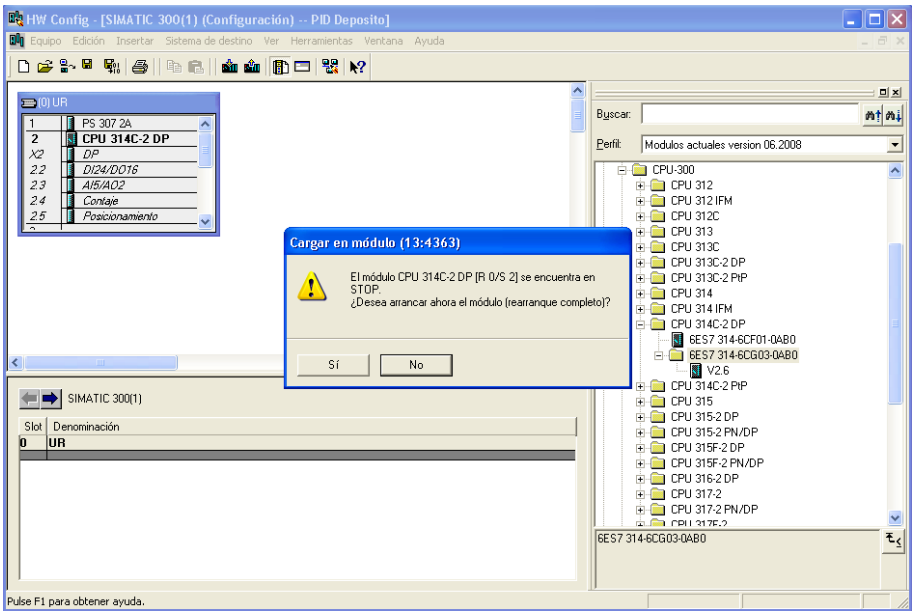

Figura 5.31. CPU modo RUN.

5.8. Programación del PLC.

Una vez acabada la configuración de la CPU y de las E/S analógicas, se vuelve a la **Figura 5.17**, en la cual se empezaría a programar las OB que hacen falta para realizar el control de la presión del depósito.

5.8.1. Alarma periódica OB35.

Para realizar la regulación continua de este proyecto, como ya se ha mencionado en algunas ocasiones, se va utilizar la función **FB41 CONT_C**. Esta será programada en la OB35 de alarma periódica, la cual se utiliza debido a su independencia con respecto al tiempo ciclo del PLC.

Las CPUs S7 ofrecen OBs de **alarmas cíclicas** o **periódicas** [9] que interrumpen la ejecución cíclica del programa en intervalos determinados. Al ajustar estos intervalos de ejecución se debe tener en cuenta que entre los eventos de arranque de las diferentes alarmas cíclicas haya tiempo suficiente para la ejecución de dichas alarmas. Para arrancarlas es necesario indicar el periodo (base de tiempo) correspondiente con STEP7 en el bloque de parámetros de alarmas cíclicas **(Figura 5.21**). En los equipos SIMATIC S7 300 la única alarma cíclica que se encuentra habilitada, de la OB30 hasta la OB38, es la OB35.

Para insertar la alarma cíclica OB35, se selecciona **Insertar**, **Bloque S7** y **Bloque de organización**, como muestra la **Figura 5.32:**

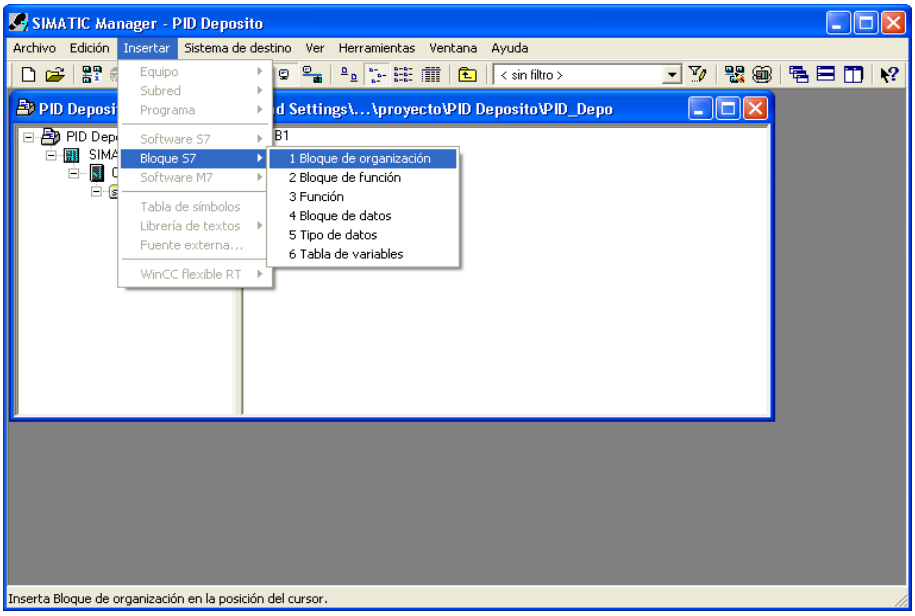

Figura 5.32. Bloque de organización

Al insertar el bloque de organización, aparecerá la ventana de **Propiedades,** en la cual hay que darle el nombre al bloque, que será OB35 y es importante elegir el lenguaje de programación, que en este caso se ha elegido el KOP (**Figura 5.33**), ya que se sigue con una mayor claridad, en comparación con los otros dos.

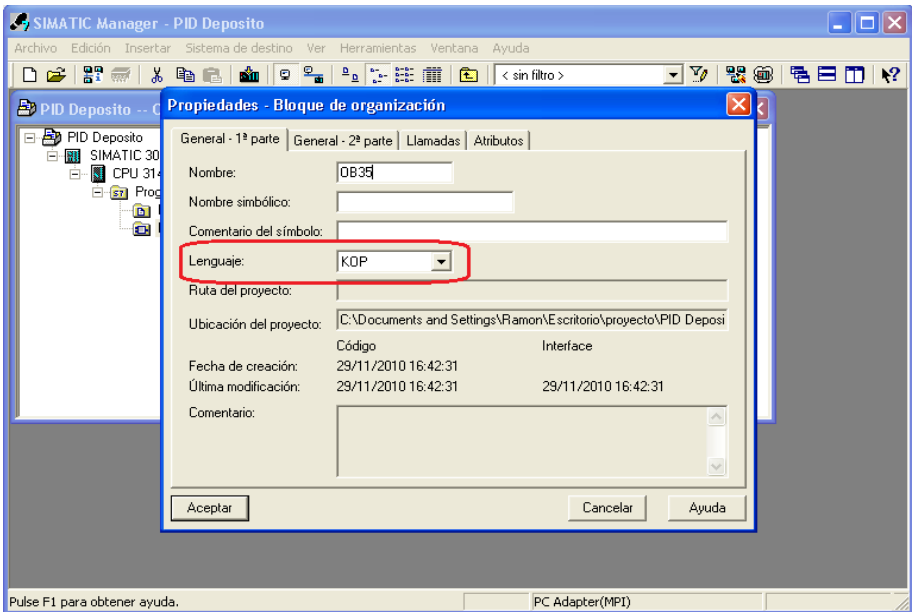

Figura 5.33. Bloque OB35.

Una vez introducido el bloque de organización, se tiene que abrir y programar dentro de este la función FB41. Esta función se encuentra en **Librerías, Standard Library**, **PID Control Blocks** (**Figura 5.34**).

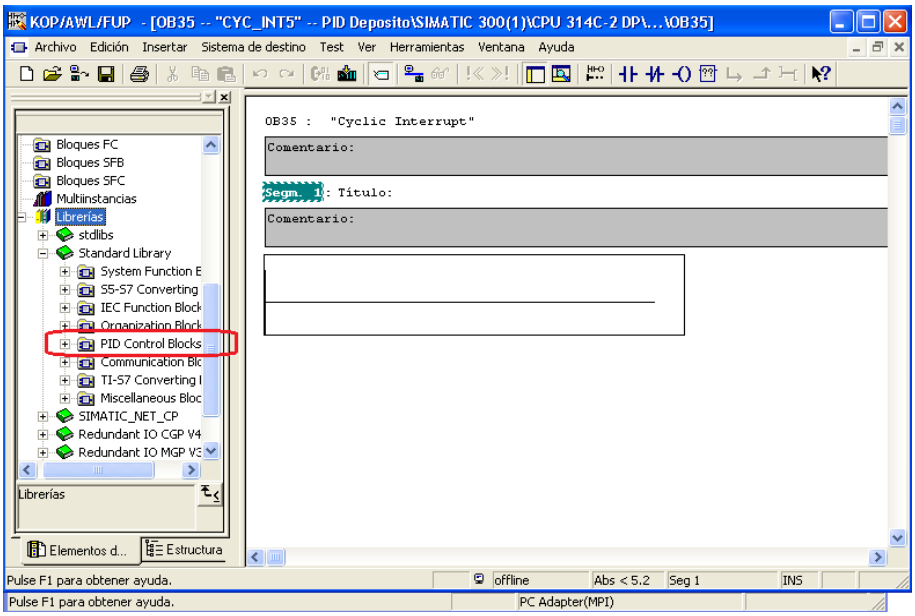

Figura 5.34. PID Control Block.

Esta librería corresponde con la función FB 41 (CONT_C), que al introducirla en la OB35 el programa la asocia a una DB, preguntando si crea la DB41, como en la **Figura 5.35**, se responderá que si y se le dará un nombre, por ejemplo, DB41.

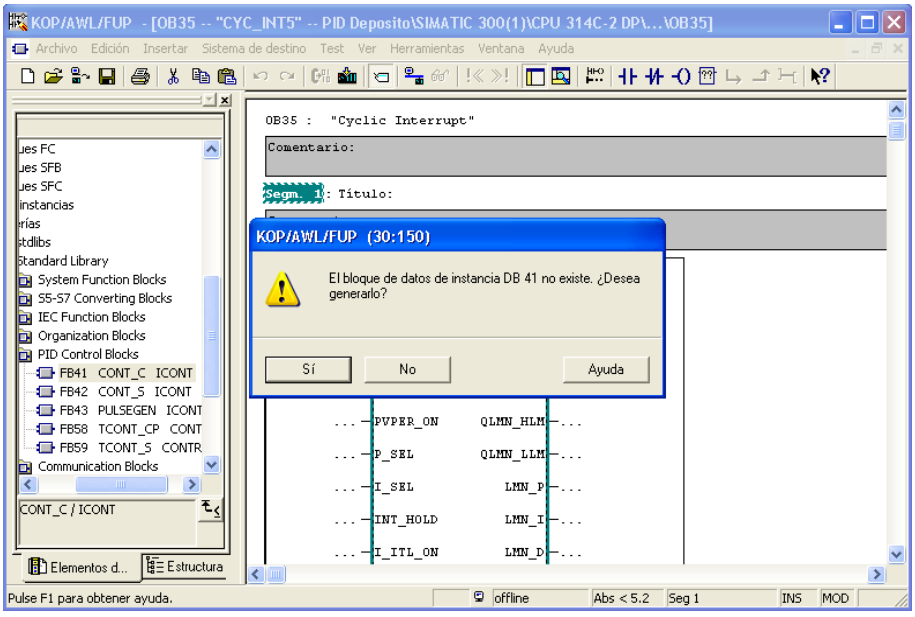

Figura 5.35. Bloque DB41.

Una vez asociada la DB, se le pondrá un nombre simbólico, para que todos los parámetros que guarda esta, aparezcan con ese nombre simbólico.

Se abre la ventana general del software (**SIMATIC Manager PID Depósito)**, en la cual aparece la DB41. Encima de esta, pulsando el botón derecho del ratón, se abre una ventana y se seleccionará **Propiedades del objeto** (**Figura 5.36**).

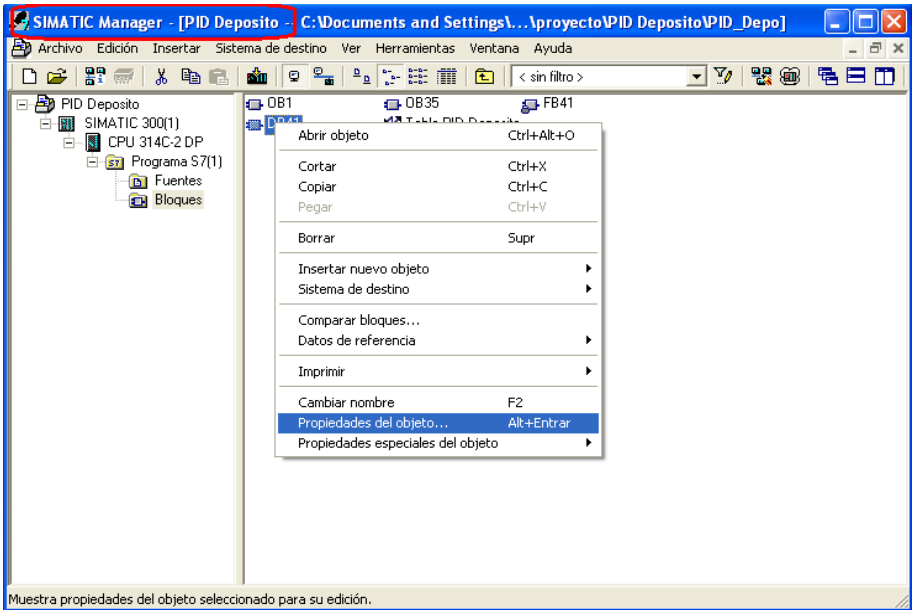

Figura 5.36.Propiedades del objeto.

Por tanto, aparecerá una ventana, **Propiedades DB de instancia del FB 41** (**Figura 5.37**), en la cual se introducirá el nombre simbólico.

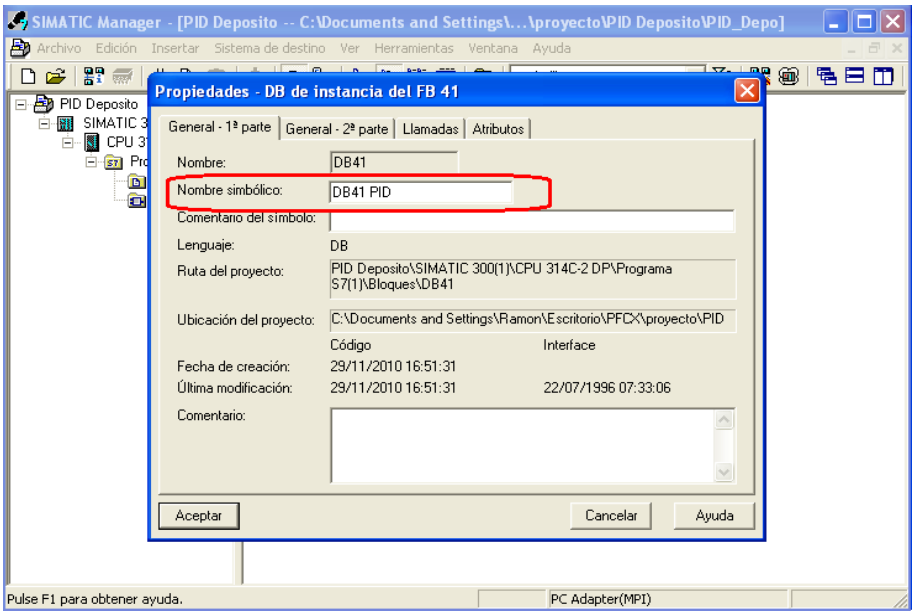

Figura 5.37. Propiedades del DB de instancia.

Una vez hecho lo anterior, se podrán asociar las señales a los parámetros adecuados de la función **FB41**, se trata de **PV_PER**, **LMN_PER** y **COM_RST** (**Figura 5.38**).

Estos parámetro constituyen la conexión entre el sistema o proceso, la tarjeta A/D y función PID del regulador. La conexión que se ha hecho en el hardware tiene que quedar reflejada a la hora de programar, por tanto:

- **PV_PER** = **PEW754** Esta es la dirección de la entrada analógica, en la cual, estará conectada el sensor, es decir la señal realimentada.
- **LMN_PER** = **PAW754** Esta es la dirección de la salida analógica, la cual, se conectará a la servovalvula, es decir, la señal de control.
- **COM_RST** = **E124.7** Esta entrada digital se utiliza como "RESET" del bloque PID. Este bit se programa en el OB100.

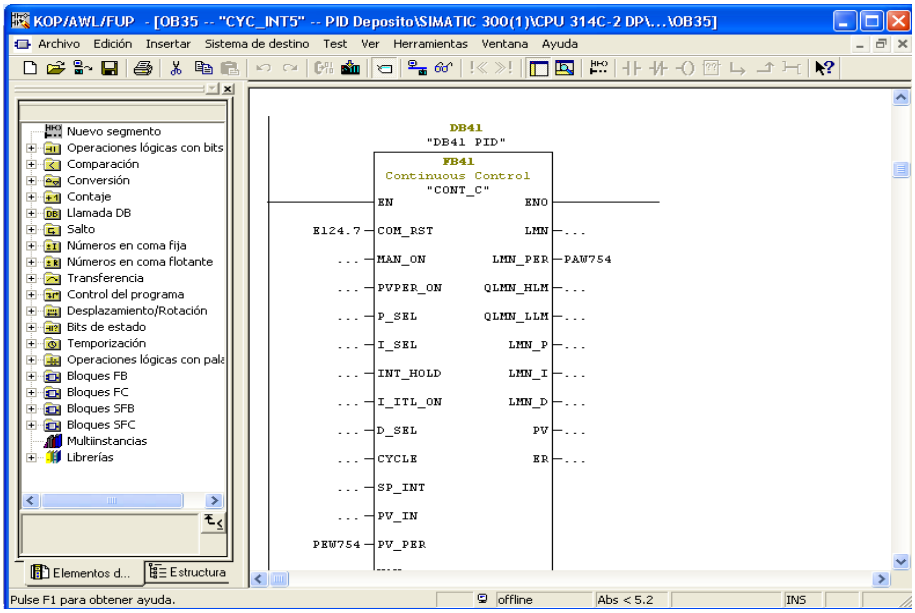

Figura 5.38. Bloque OB35.

5.8.2. Reset del regulador.

Se ha introducido en este proyecto reset externo del regulador, utilizando la entrada digital **E124.7**. Como se ha mencionado anteriormente, esta acción se programará en la OB100, que es de rearranque.

Al activar el parámetro **COM_RST** a través de la entrada digital se producirá el rearranque y la inicialización de los valores de salida del PID. Esta acción se debe hacer en el paso de STOP a RUN del PLC, si no se reinicia el PID a la par que la CPU, este ira acumulando errores de regulación cada vez que se reinicie y hasta que no desaparezca este error acumulado, después de cierto tiempo, la regulación no estará dentro de los márgenes específicos del sistema para el que fue diseñado. Para evitar esto, se debe utilizar la OB100, que reseteará el PLC después de reiniciar el PID.

El procedimiento es el mismo que al insertar el OB35, se selecciona **Insertar**, **Bloque S7** y **Bloque de organización** (**Figura 5.32**), luego aparecerá la ventana de **Propiedades,** en la cual hay que darle el nombre al bloque, que será OB100 y el lenguaje será el KOP. En este caso no se le dará un nombre simbólico.

En el bloque OB100 se insertará la función FB41 (**Figura 5.34**), pero esta vez solo se va a tener en cuenta la entrada **COM_RST**, en la que se escribirá la entrada digital **E124.7**. Todo esto quedará como se ve en la **Figura 5.39**.

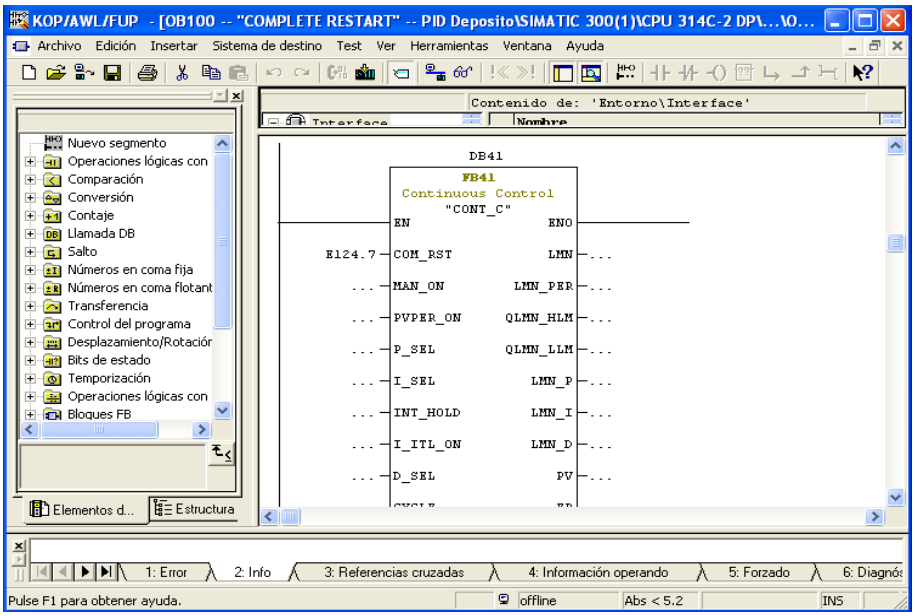

Figura 5.39. Reset.

Una vez se ha terminado de programar todos los bloques que se requieren para la regulación del sistema, se transferirán todos estos datos a la CPU y se comprobará que no da ningún error. Como muestra la **Figura 5.40**, hay que presionar el botón resaltado para cargar los datos.

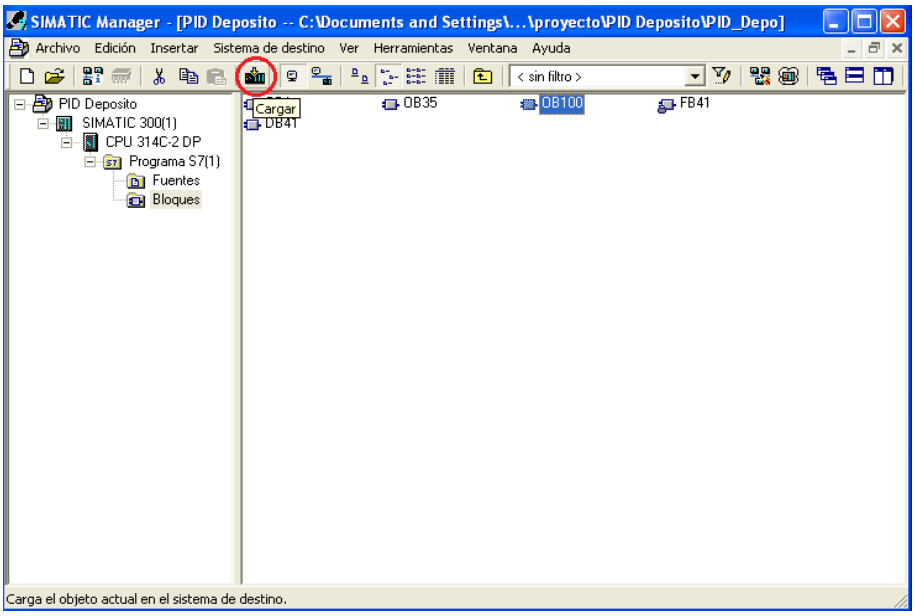

Figura 5.40. Cargar datos.

5.9. Tabla de variables

Una vez programado el PLC y cargado todos estos datos en su memoria, se tendrá que tener una conexión con la **DB41**, en la cual se podría modificar y verificar los valores de esta de forma mucho más fácil que con la propia DB. Para lograr esto se creará una tabla de variables, en la que podrían aparecer todos los parámetros que se necesiten supervisar o forzar.

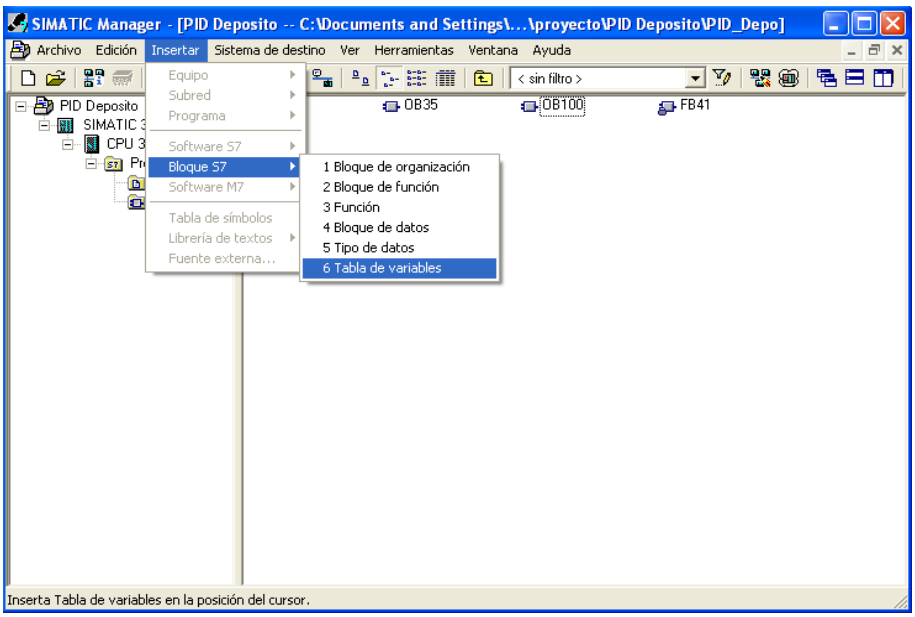

Figura 5.41. Tabla de variables.

Para crear la tabla se selecciona **Insertar**, **Bloque S7** y **Tabla de variables** (**Figura 5.41**), luego aparecerá la ventana de **Propiedades** (**Figura 5.42**), en la que se le dará el nombre simbólico de **Tabla PID Depósito.**

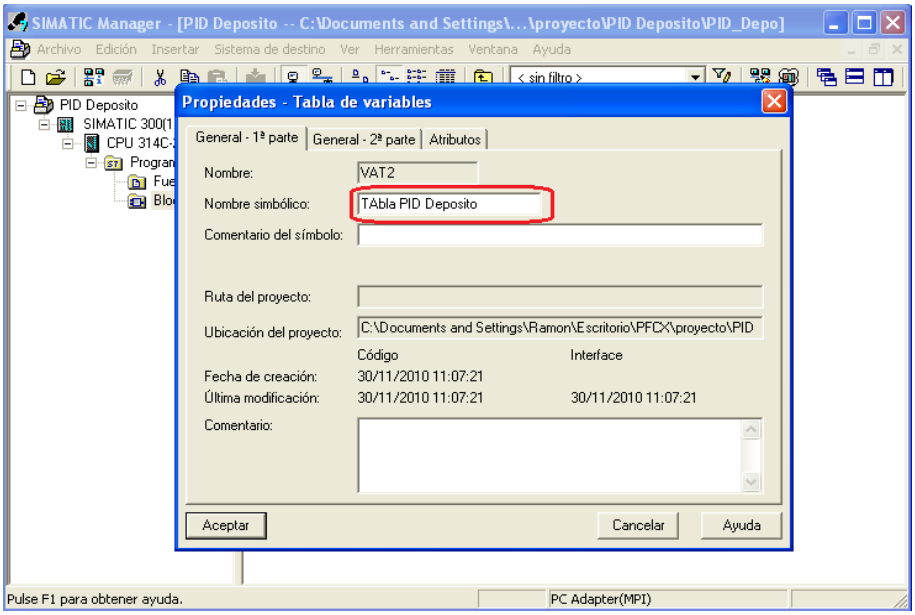

Figura 5.42. Propiedades de la tabla.

Poner nombres simbólicos no es obligatorio, pero ayudará mucho a la hora de utilizar la tabla de variables, ya que se sabrá de que DB se estará forzando o supervisando los datos.

Al terminar en la ventana de propiedades, aparecerá la tabla de variables junto a la FB41 y las OB, como se ve en la **Figura 5.43**. Todos estos elementos se encuentran ya programados y transferidos a la CPU del PLC.

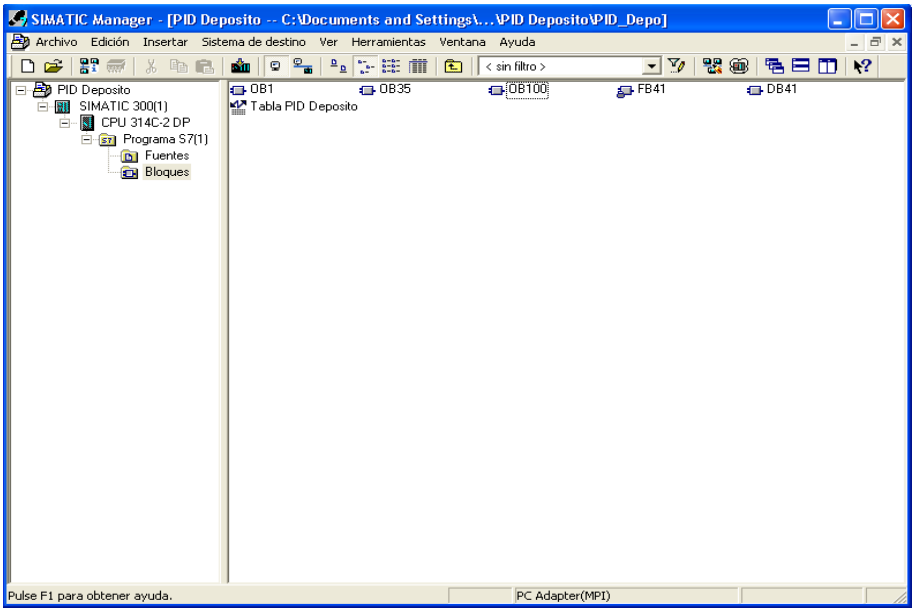

Figura 5.43. Elementos de programación.

El siguiente paso es el de introducir en la tabla de variables todos los parámetros que se necesiten, o que sean importantes en la regulación del sistema. En este caso se ha introducido los parámetros de la función FB 41 (CONT_C) que se ven en la **Figura 5.44** y **Figura 5.45.**

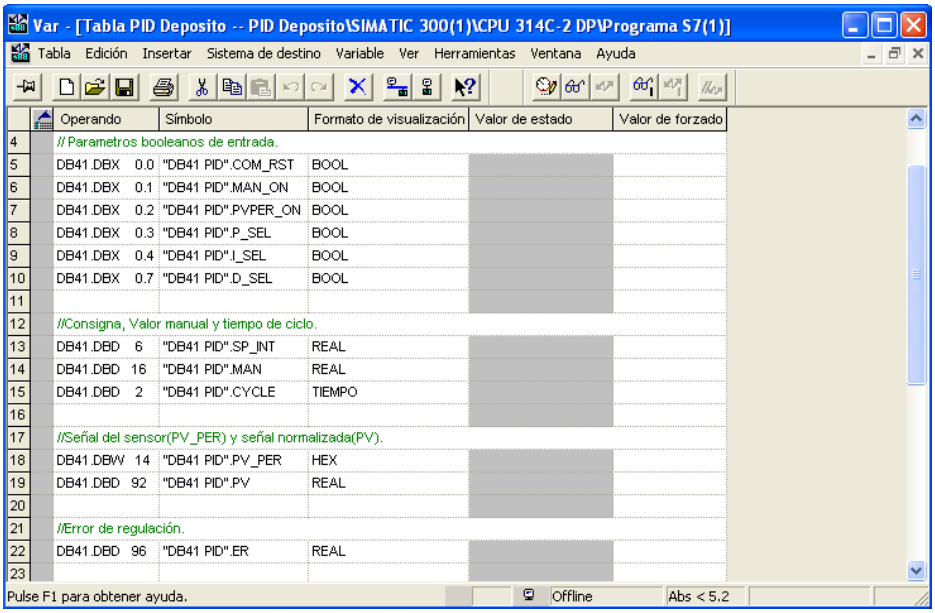

Figura 5.44. Parámetros de la función FB41.

Se puede observar en la **Figura 5.44** y **Figura 5.45** en la que se introducen los parámetros con el nombre simbólico de la **DB41 PID**.

Se completará la tabla teniendo en cuenta el nombre simbólico de la DB y el tipo de dato de cada parámetro.

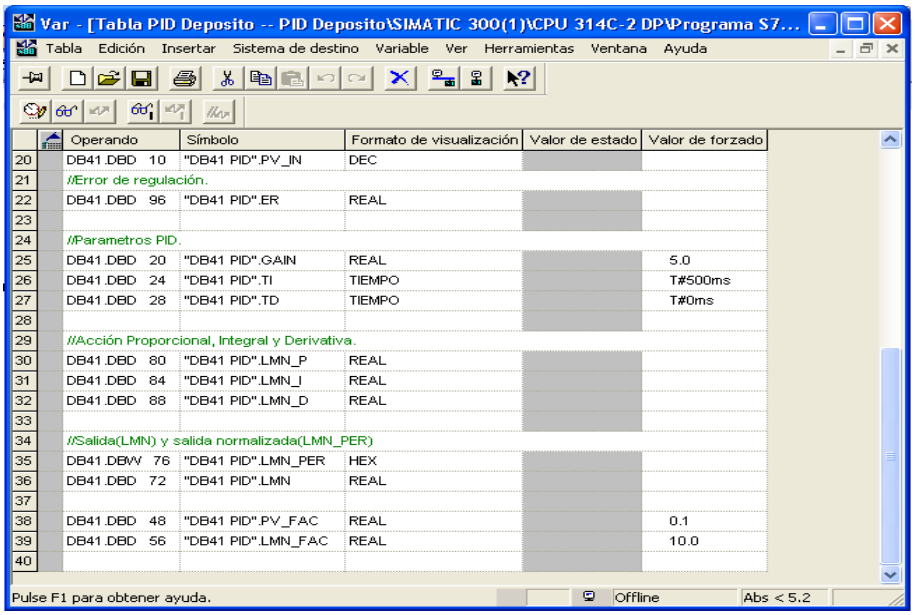

Figura 5.45. Otros parámetros de la función FB41.

5.10. Parametrización del regulador PID.

La parametrización del regulador se podría realizar visualizando la OB35 o la tabla de variables creada, pero el entorno STEP7 ofrece una opción sencilla y grafica para realizar la parametrización. Este entorno grafico se puede ejecutar siguiendo los pasos de la **Figura 5.46.**

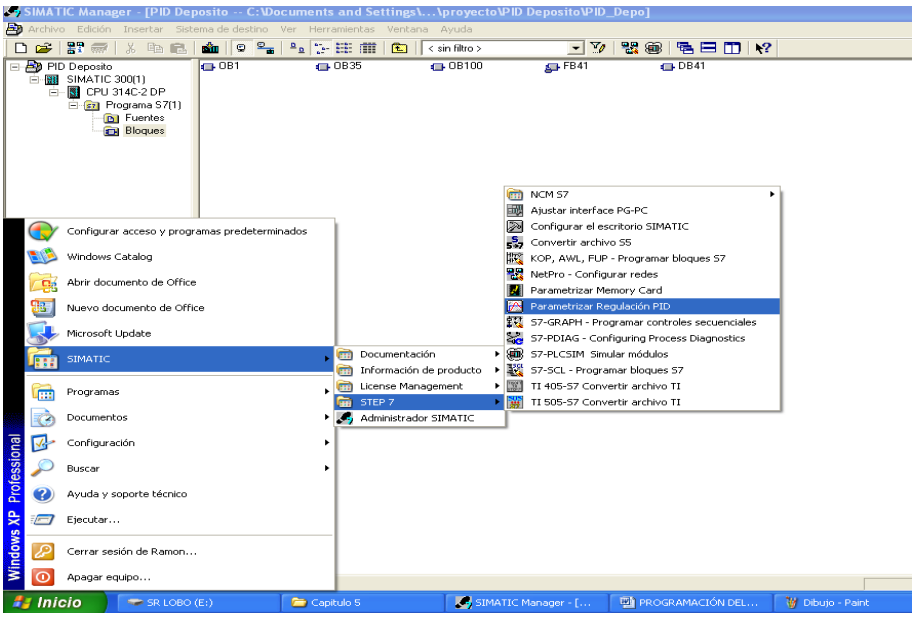

Figura 5.46. Parametrización del regulador PID.

Con opción que ofrece STEP7, se pueden modificar los elementos más importantes de la función FB41. Este entorno no permite cambiar la consigna (**SP_INT**) ni el valor asignado en modo manual (**MAN**), pero ofrece una visualización grafica de las variables de proceso. Es por esto que a la hora de realizar el control del sistema, se utilizara tanto la tabla de variables como este entorno grafico.

Una vez seleccionado "**Parametrizar Regulación PID**", aparece la ventana de la **Figura 5.47**, en la que hay que ir a la pestaña de **Archivo**, **Abrir** y aparecerá la ventana de **Figura 5.48**, en la cual se debe seleccionar el modo **Online** y abrir la DB a la que está asociada la **FB41**, que en este caso es la **DB41.**

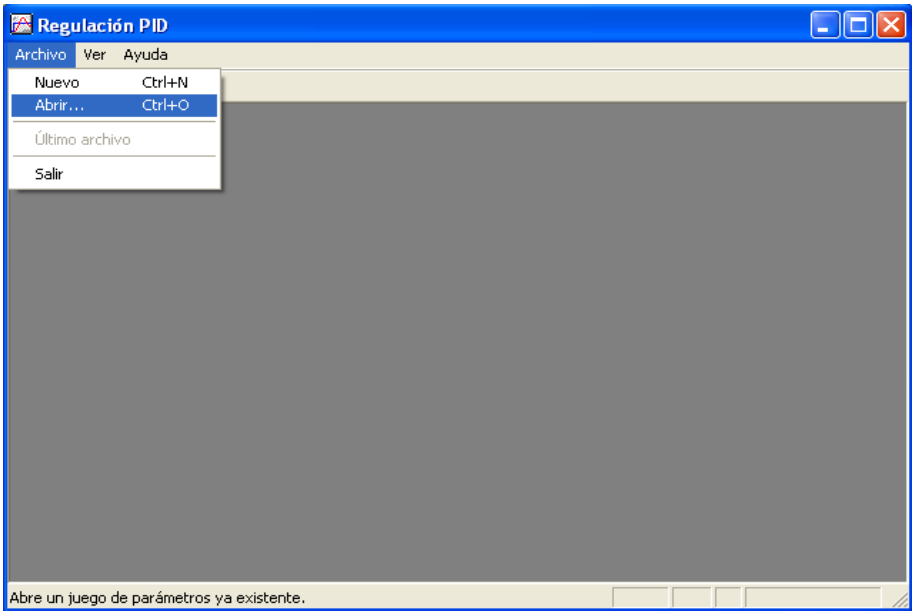

Figura 5.47. Regulación PID.

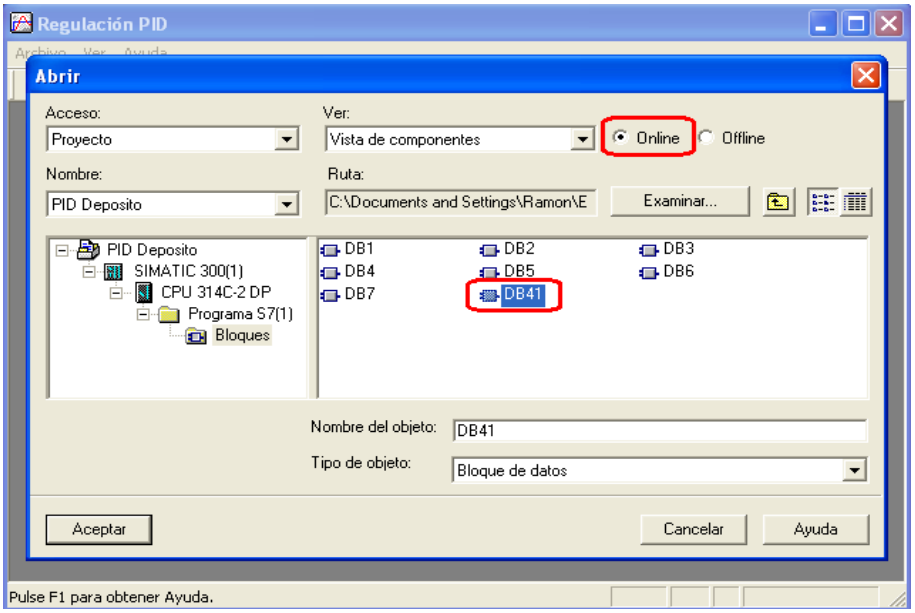

Figura 5.48. Diagrama de bloques DB41.

Una vez acabado con lo anterior, aparecerá la ventana de la **Figura 5.49**, en la cual se podrán parametrizar los valores más significativos.

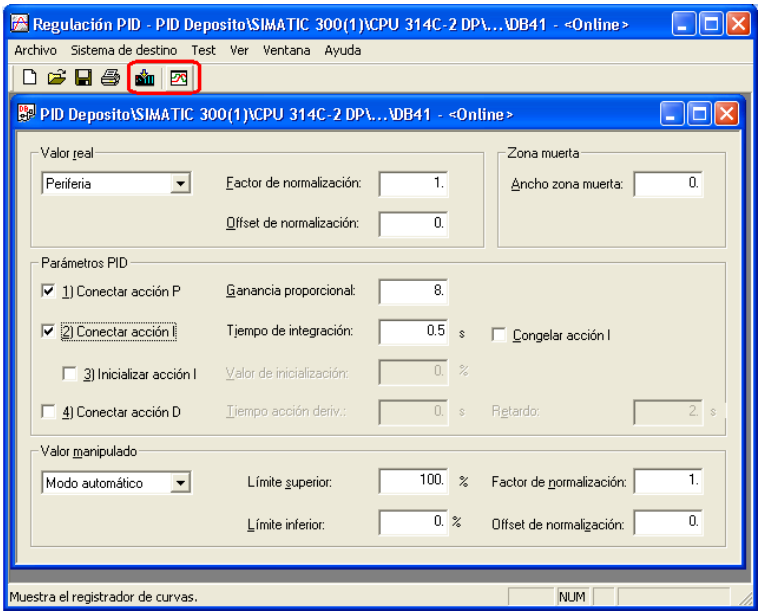

Figura 5.49. Regulación PID Online.

Este entorno está relacionado con la **DB41** a través de los siguientes parámetros:

- Periferia: Se lee la entrada PV_PER.
- Interno: Se lee de la entrada PV_IN.

- Factor de normalización: PV_FAC.
- Offset de normalización: PV_OFF.
- Ancho de zona muerta: DEADB_W.

- Conectar acción P: P_SEL.
- Ganancia proporcional: GAIN.

- Conectar acción I: I_SEL.
- Tiempo de integración: Ti.
- Congelar acción I: INT_HOLD.
- Inicializar acción I: I_ITL_ON.
- Valor de inicialización: I_ITLVAL.

- Conectar acción D: D_SEL.
- Tiempo de acción derivativa: Td.
- Retardo: TM_LAG.

- Modo automático: si MAN_ON = 0.
- Modo manual: si MAN_ON = 1

- Límite superior: LMN_HLM.
- Límite inferior: L MN_LLM.

- Factor de normalización: LMN_FAC.
- Offset de normalización: LMN_OFF.

Una vez que se ha decidido que valores han de tomar dichos parámetros hay que cargarlos en la CPU para luego ejecutar el **registrador de curvas**. En la **Figura 5.49** están resaltados estos botones

Cada vez que se quiera modificar los parámetros en este entorno grafico, se debe parar la ejecución y cargar los nuevos valores en la CPU. Este es otro motivo de la necesidad de la tabla de valores, en la cual no hace falta detener la ejecución.

6. RESULTADOS OBTENIDOS.

6.1. Introducción.

En este capítulo se muestra un análisis de las capacidades del sistema, las reacciones de este ante las posibles situaciones a las que pueda ser sometido por parte del usuario y su respuesta ante ellas. Se muestran unas pruebas finales que demuestran que el sistema es capaz de funcionar correctamente en todo momento.

Se configurará el regulador como P, PI y PID, pudiéndose así comprobar los efectos de los valores Kp, Ti y Td que ya se han explicado teoricamente en el capitulo anterior.

En los sistemas reales, en los cuales no es fácil obtener su modelo matemático, cuando los PID se realizan con software, se puede usar un software especial para sintonizar los parámetros Kp, Td y Ti (PID self-tuner). En otros casos la regulación se realiza con tarjetas especiales que llevan definidas unas funciones para realizar la auto-sintonía.

Exceptuando los casos antes mencionados, la sintonización se puede realizar utilizando otros métodos sin necesidad de software que los sintonice. Se pueden utilizar, entre otros, "**Ziegler Nichols**" o "**Harriot**". Para este documento se ha optado por utilizar el método de **Ziegler Nichols**, el método de las oscilaciones sostenidas.

6.2. Reglas de sintonización Ziegler-Nichols.

La **Figura 6.1** muestra el control PID de una planta [13]. Si se puede obtener un modelo matemático de la planta, es posible aplicar diversas técnicas de diseño con el fin de determinar los parámetros del controlador que cumpla las especificaciones en estado transitorio (régimen transitorio) y en estado estable (régimen permanente) del sistema en lazo cerrado. Sin embargo, si la planta es tan complicada que no es fácil obtener su modelo matemático, tampoco es posible un enfoque analítico para el diseño de un controlador PID. En este caso, debemos recurrir a los enfoques experimentales para la sintonización de los controladores PID.

El proceso de seleccionar los parámetros del controlador que cumplan con las especificaciones de desempeño se conoce como sintonización del controlador. Ziegler y Nichols sugirieron más reglas para sintonizar los controladores PID (lo cual significa establecer valores **Kp, Ti** y **Td**) con base en las respuestas escalón experimentales o basadas en el valor de **Kp** que se produce en la estabilidad marginal cuando sólo se usa la acción de control proporcional.

Las reglas de **Ziegler-Nichols**, que se presentan a continuación, son muy convenientes cuando no se conocen los modelos matemáticos de las plantas. (Por supuesto, estas reglas se aplican al diseño de sistemas con modelos matemáticos conocidos.)

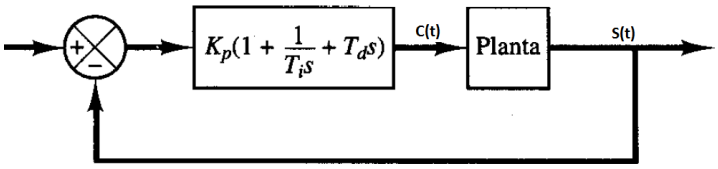

Figura 6.1. Control PID de una planta.

Ziegler y Nichols propusieron unas reglas para determinar los valores de la ganancia proporcional **Kp,** del tiempo integral **Ti** y del tiempo derivativo **Td,** con base en las características de respuesta transitoria de una planta específica. Tal determinación de los parámetros de los controladores PID o de la sintonización de los controles PID la realizan los ingenieros mediante experimentos sobre la planta.

Existen dos métodos denominados reglas de sintonización de Ziegler-Nichols. En ambos se pretende obtener un 25% de sobrepaso máximo en la respuesta escalón (véase la **Figura 6.2**). El primer método se basa en la curva de respuesta del sistema en lazo abierto y el segundo se basa en la oscilación del sistema en lazo cerrado.

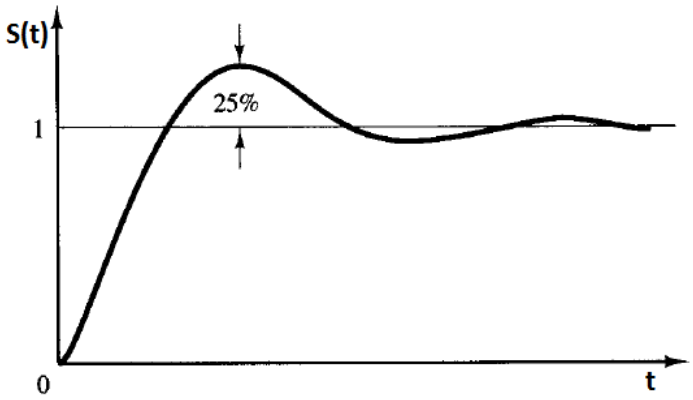

Figura 6.2. Curva de respuesta escalón unitario con sobrepaso máximo de 25%.

6.2.1. Método de oscilación mantenida.

En este método [13], primero se establece **Ti** = ∞ y **Td** = 0. Usando sólo la acción de control proporcional (véase la **Figura 6.3**), se incrementa **Kp** de 0 a un valor crítico **Kc**, en donde la salida exhiba oscilaciones sostenidas. (Si la salida no presenta oscilaciones sostenidas para cualquier valor que pueda tomar **Kp,** no se aplica este método.) Por tanto, la ganancia crítica **Kc,** y el periodo **Pcr** correspondiente se determinan experimentalmente (véase la **Figura 6.4**). Ziegler-Nichols sugirieron que se establecieran los valores de los parámetros **Kp**, **Ti** y **Td** de acuerdo con la fórmula que aparece en la **Tabla 6.1**.

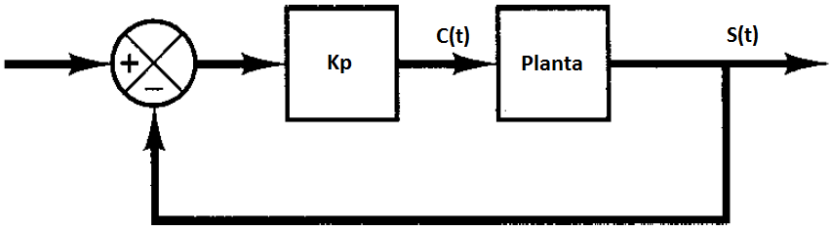

Figura 6.3. Sistema en lazo cerrado con control proporcional.

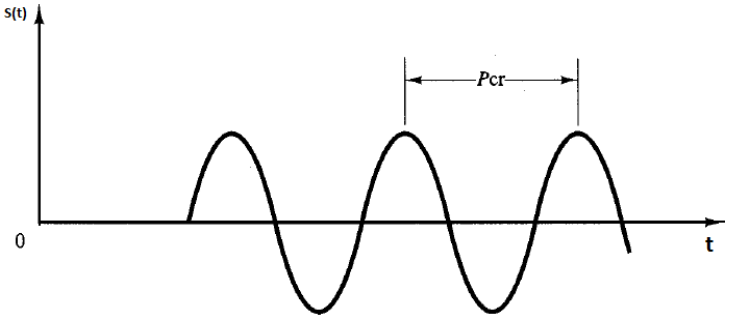

Figura 6.4. Oscilación sostenida con periodo Pcr.

Tipo de controlador	Kp	Ti	Td
P	0.5Kcr	∞	0
PI	0.45Kcr	$\frac{1}{1.2}$ Pcr	0
PID	0.6Kcr	0.5Pcr	0.125Pcr

Tabla 6.1. Regla de sintonización de Ziegler-Nichols.

Las reglas de sintonización de Ziegler-Nichols (y otras reglas de sintonización que se presentan en la literatura) se han usado ampliamente para sintonizar controladores PID en los sistemas de control de procesos en los que no se conoce con precisión la dinámica de la planta. Tales reglas de sintonización han demostrado ser muy útiles durante muchos años. Por supuesto, las reglas de sintonización de **Ziegler-Nichols** se aplican a las plantas cuya dinámica se conoce. (En estos casos, se cuenta con muchos enfoques analíticos y gráficos para el diseño de controladores PID, además de las reglas de sintonización de Ziegler-Nichols).

6.3. Sintonía de los parámetros PID.

Como se ha mencionado en la sección anterior, a continuación se desconectarán las acciones integral y derivativa y también se asignará el valor 0 a Kp. Por tanto se abrirá la tabla y los parámetros **I_SEL** y **P_SEL** se desconectaran y al parámetro **GAIN** se le asignará el valor 0.

Por otra parte, se recuerda que los valores de las entradas y salidas del PLC son porcentajes (Ver sección 5.5.2.1.). Existe la opción de cambiar estos rangos con los parámetros **PV_FAC**, **PV_OFF**, **LMN_FAC** y **LMN_OFF**.

En este proyecto se ha trabajado en bares, no en porcentajes. Es por eso que para las expresiones (5.12) y (5.13) se han elegido los valores siguientes:

PV_FAC= 0.1

LMN_FAC=10

Hay que tener especial cuidado de elegir el rango ya que para valores distintos las acciones del controlador van a variar. Un ejemplo sería el de la ganancia, ya que en el caso de que se eligiese LMN_FAC = 1, la ganancia seria 10 veces mayor.

Para observar la salida se ejecutará el entorno grafico que ofrece STEP7 (**Figura 6.5**).

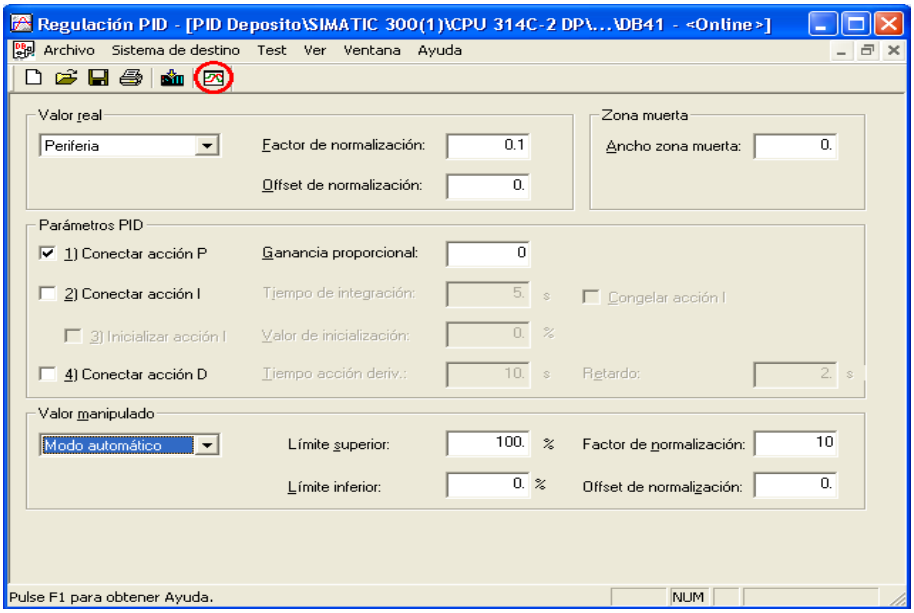

Figura 6.5. Entorno grafico PID.

Se ejecuta el registrador de curvas (**Figura 6.5)** y activando el botón "ajustes" (**Figura 6.6**), podremos visualizar cuatro de entre todos los valores que podamos tener en el lazo que estamos regulando, y podemos configurar el color, límite, rango, tiempo de visualización y velocidad de recepción de cada uno.

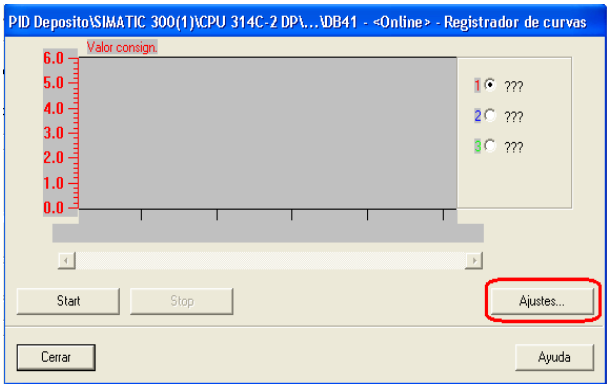

Figura 6.6. Registrador de curvas.

En la ventana de ajustes solo se activará el parámetro **consigna** y **valor real** y se le han asignado los colores rojo y azul respectivamente. También se ha asignado un rango más pequeño de amplitud **(Figura 6.7**).

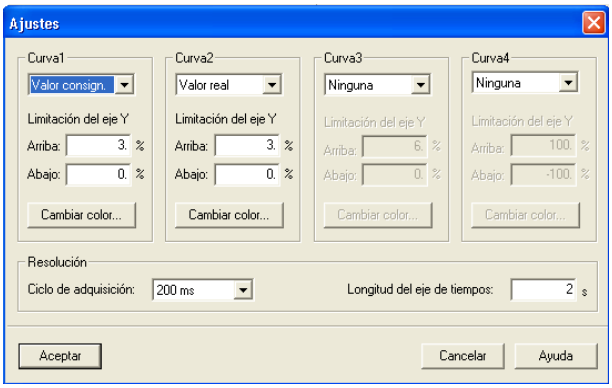

Figura 6.7. Ajustes.

Una vez ajustado el registrador de curvas; se irá incrementando el valor de **Kp** (**GAIN)** en la tabla de valores hasta que se llegue a obtener en la salida oscilaciones sostenidas. Lo anterior se verá reflejado en las siguientes figuras:

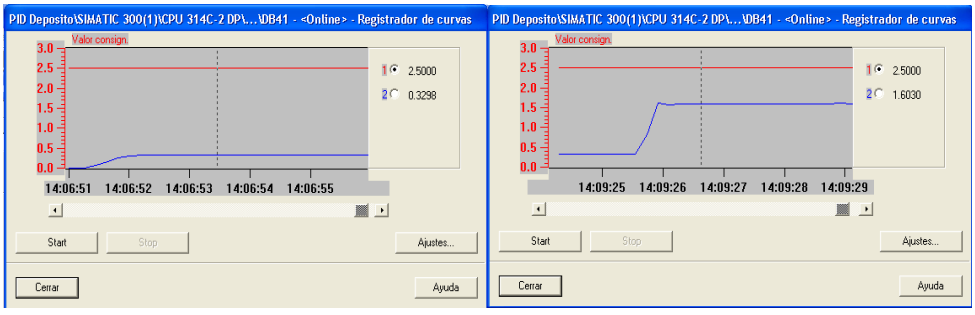

Figura 6.8. Kp = 2 y Kp = 5.

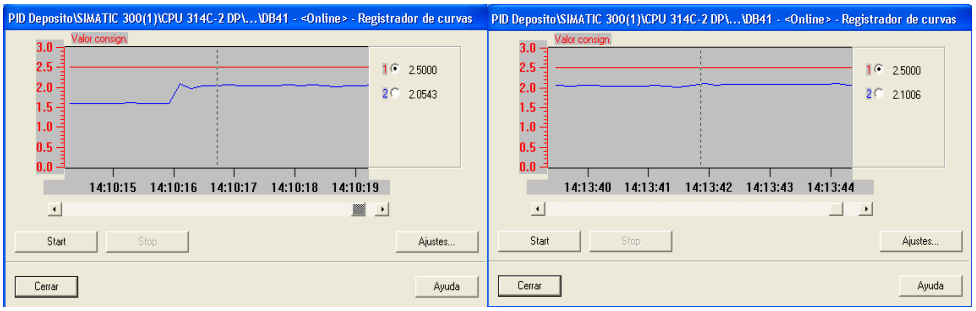

Figura 6.9. Kp = 10 y Kp = 11.

Se puede observar, al intentar hacer oscilar el sistema, los efectos de un regulador de tipo P, es decir, que el **valor real** se va a acercar en mayor o menor medida (dependiendo del valor de Kp) a la **consigna** pero nunca la alcanzará (ver **Figura 6.8** y **Figura 6.9**), es decir, que siempre existirá un error.

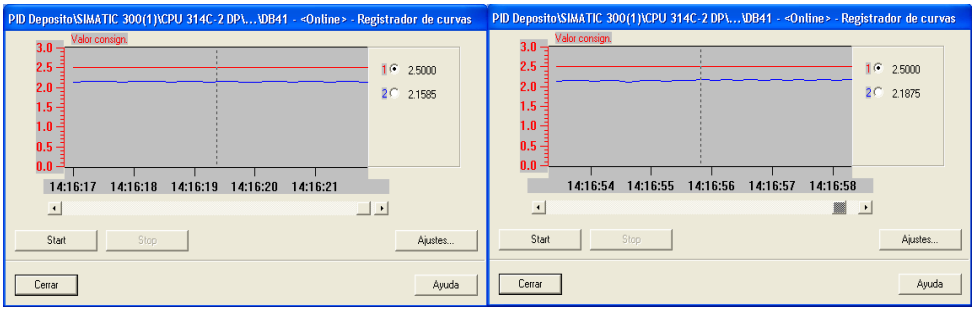

Figura 6.10. Kp = 13 y Kp = 14.

Se puede observar en la **Figura 6.10** que el **valor real** ya no presenta cambios significativos al variar la ganancia, sin embargo, sí que presenta cierta oscilación.

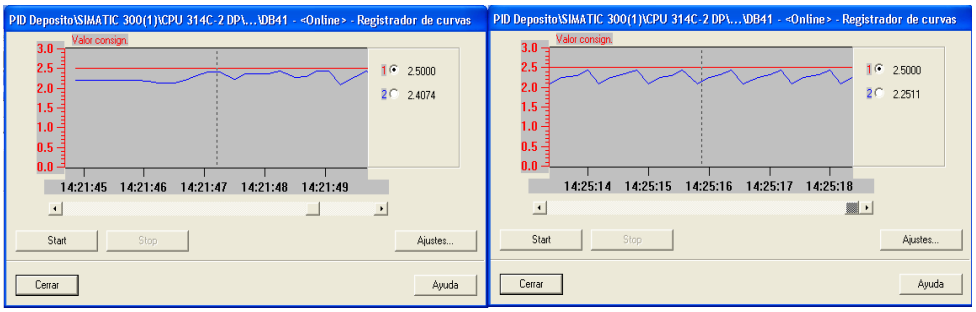

Figura 6.11. Kp = 15.

Como era de prever, al aumentar el valor de la ganancia (ver **Figura 6.11**) a 15 el sistema empieza a oscilar y mantiene esta inestabilidad. Por tanto la ganancia critica (Kcr) será de 15 y el periodo crítico (Pcr) será de aproximadamente 800ms.

Para los parámetros críticos hallados los valores de sintonización serian los siguientes:

Tipo de controlador	Kp	Ti(ms)	Td(ms)
P	7,5	∞	0
PI	6,75	667	0
PID	9	400	100

Tabla 6. 2. Valores de los parámetros PID.

6.4. Efecto del regulador P.

Como se ha visto en la **sección 6.3**, un regulador de tipo P no podrá controlar completamente el sistema ya que siempre existirá un error, por muy grande que sea la Kp. Aunque como se ha visto para una alta ganancia el sistema puede presentar problemas de inestabilidad.

A continuación se verá cómo responde el sistema con diferentes valores de **consigna** para el valor de Kp hallado por el método de sintonización de Ziegler-Nichols. En este caso también se ha incluido en la grafica el valor del **error de regulación** y se ha cambio la amplitud de la grafica. Este tipo de regulador corresponde con la expresión:

$$C(s) = K_p \varepsilon(s) = 7{,}5\varepsilon(s)$$

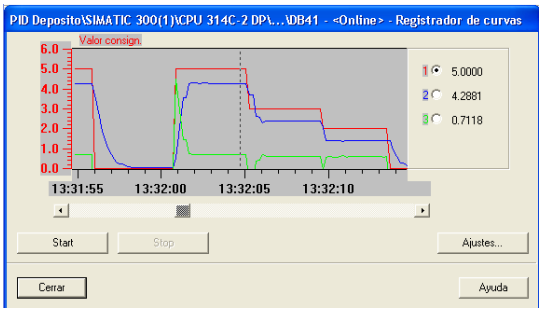

Figura 6.12. Acción P experimento 1º.

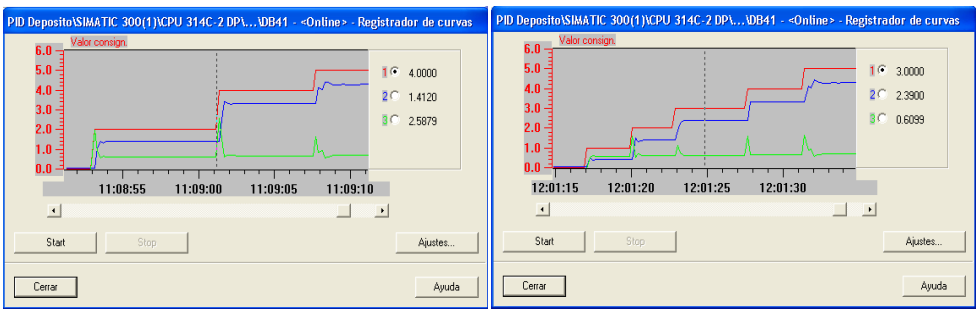

Figura 6.13. Acción P experimento2º.

Se ve en las **Figura 6.12** y **Figura 6.13** cómo será necesario incluir otra acción que elimine este error en régimen permanente. También cabe destacar que este tipo de sistemas presentan mucho ruido sobre todo presentes al cambiar la **consigna**, aunque también se ve en se pueden ver en régimen permanente**.**

6.5. Regulador PI.

Es la estructura más usual del controlador. En esta ocasión probaremos el efecto de la acción integral I (**Figura 6.14** y **Figura 6.15**), activando el parámetro **I_SEL** de la tabla de variables (I_SEL=1). Así pues, a partir de ahora trabajaremos con el regulador del tipo PI, que es el adecuado para este proceso. De este modo, cuando haya cambios de consigna, el régimen transitorio se hace notar la acción proporcional, mientras que en el régimen permanente el error desaparecerá por completo a causa de la acción integral.

En este caso la expresión del regulador, incluyendo los valores de Ziegler-Nichols, sería la siguiente:

$$C(s) = e(s)K_p\left(1+\frac{1}{T_iS}\right) = e(s)6,75\left(1+\frac{1}{667S}\right)$$

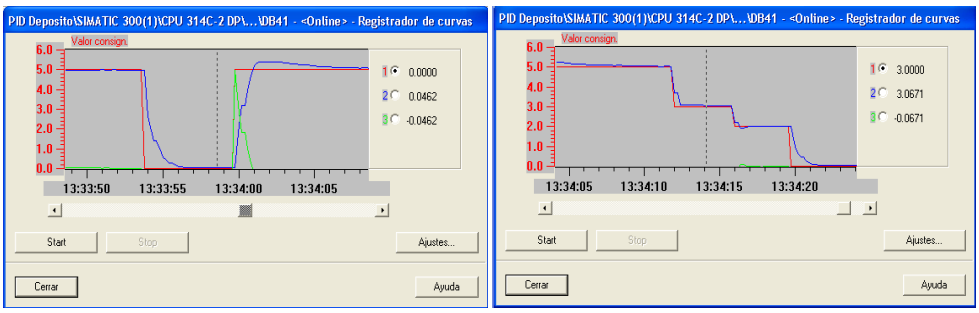

Figura 6.14. Acción PI experimento 1º.

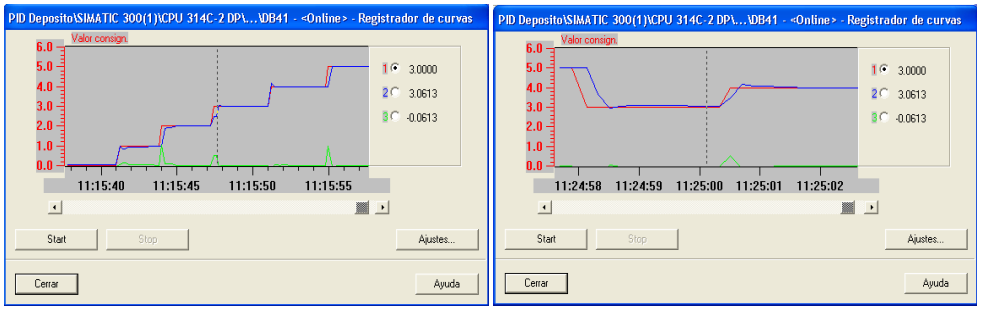

Figura 6.15. Acción PI experimento 2º.

Se observa como el regulador de tipo PI mejora notablemente el proceso a regular, eliminando totalmente el error en régimen permanente, a pesar de las perturbaciones que presentan este tipo de sistemas.

6.6. Regulación PID.

Ahora se incluirá la acción D al regulador con el fin de ver gráficamente la influencia de esta acción. Como se ha mencionado en la **sección 5.3.3**, la acción derivativa tiene en cuenta la variación del error, es decir, es sensible al ruido. Por tanto en presencia de altos niveles de ruido se debe limitar esta acción o prescindir de ella. La acción derivativa más que una mejora en esta situación es un problema ya que amplifica el ruido existente.

En este caso la expresión del regulador, con los valores de Ziegler-Nichols calculados, sería la siguiente:

$$C(s) = e(s)K_p\left(1+\frac{1}{T_iS}+T_DS\right) = e(s)9\left(1+\frac{1}{400S}+100S\right)$$

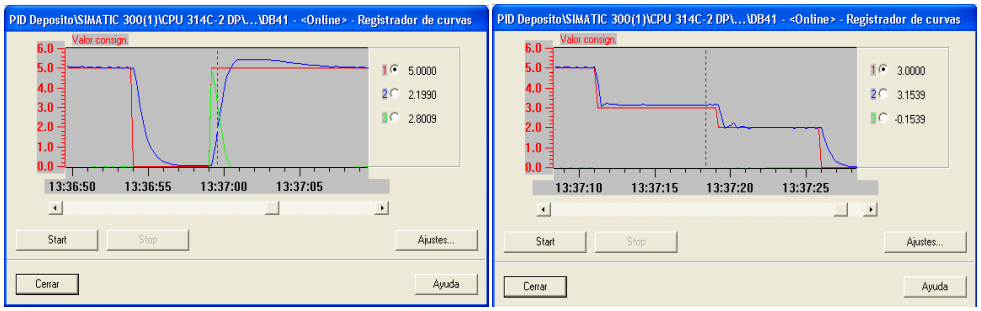

Figura 6.16. Acción PID experimento 1º.

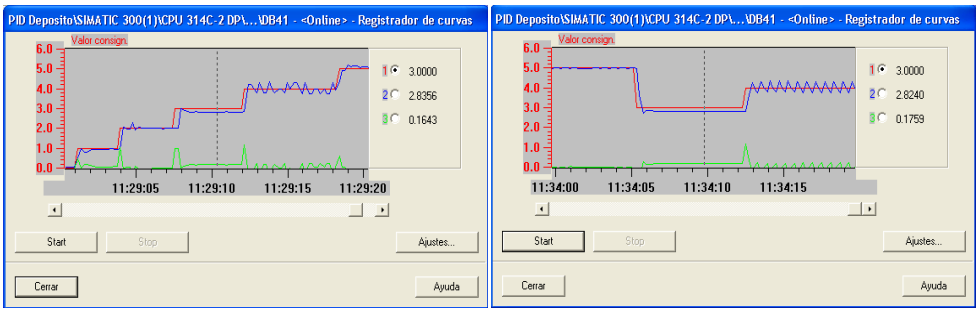

Figura 6.17. Acción PID experimento 2º.

Se puede observar en la **Figura 6.16** y **Figura 6.17** como al incluir la acción derivativa al regulador se vuelve inestable. Esta inestabilidad es por el ruido existente en el sistema, que se produce

por a las turbulencias del aire a presión y también a las vibraciones producidas en los diferentes elementos neumáticos.

Se puede concluir que en este caso la acción derivativa se debería desconectar si se quiere controlar de manera adecuada la presión del acumulador neumático, ya que en este caso no incluye ninguna mejoría al sistema.

7. CONCLUSIONES Y TRABAJOS FUTUROS.

7.1. Conclusiones.

El objetivo general de este proyecto es el de controlar la presión de un deposito neumático a través de la función PID **FB41 (CONT_C)** que lleva integrada el software de programación **STEP7** de **SEIMENS.**

En primer lugar se diseñó y se realizó el montaje del circuito neumático, el cual no supuso una gran dificultad ya que consta de pocos elementos y es relativamente sencillo.

Una vez constituido el circuito neumático, se realizó el cableado de las entradas y salidas del PLC con el circuito neumático. Luego se procedió a parametrizar estas E/S en el PLC, para que funcionaran según las necesidades del sistema neumático.

En último lugar, se procedió a realizar una correcta programación y parametrización de la función **FB41**. Esta parte dificultó en algunos momentos la realización del proyecto, ya que se desconocían muchos aspectos del funcionamiento interno del autómata.

El trabajo desarrollado en este proyecto, pese a ser muy específico y estar englobado en el uso de tecnología neumática, permite desarrollar conocimientos muy generalistas sobre las técnicas utilizadas actualmente para llevar a cabo proyectos de control de procesos industriales, bancos de pruebas y plantas industriales.

Desde un primer momento se tenía claro que la regulación se debía realizar mediante un PID, pero se desconocía cómo programarla en el autómata. Esta información se obtuvo de diferentes manuales, especialmente de los que pone a disposición del usuario la casa Siemens.

Por último, como objetivo docente, a lo largo de la carrera se aprenden los usos de la automatización industrial y se aprende a abordar proyectos sencillos de automatización, pero no se profundiza. Todo lo contrario sucede con los controladores PID que se estudian en profundidad en diversas asignaturas relacionadas con la ingeniería de control y sistemas y se aprende a implementar su uso en programas como Matlab Simulink que tienen un lenguaje esencialmente matemático, pero no se desarrolla su implementación a nivel de lenguaje de instrucciones o contactos.

Igualmente se estudia el lenguaje de programación sin entrar en detalle sobre la existencia de bloques de funciones que permiten realizar estas funciones de regulación PID. Por tanto, ha sido a raíz de este proyecto cuando he podido trabajar con estos bloques de funciones que amplían enormemente las ventajas de trabajar con los autómatas programables.

7.2. Trabajos futuros.

Como futuros desarrollos siguiendo la línea del proyecto, se proponen una serie de ampliaciones destinadas a mejorar el presente trabajo.

Se podría desarrollar un sistema que fuese capaz de detectar fugas en el circuito neumático realizando un cálculo diferencial del flujo entre dos puntos de la instalación como podrían ser a la salida del filtro regulador y en la entrada del acumulador neumático, teniendo siempre en cuenta la cantidad de aire que se expulsa al vacío. Si se detectase cualquier tipo de fuga el sistema debería avisar al usuario y si fuese preciso incluso detener la entrada de aire mediante una válvula de simple efecto normalmente cerrada colocada a la entrada de aire del circuito.

Para realizar el proyecto el departamento nos proporcionó además del material neumático, un ordenador y un PLC compuesto por una fuente de alimentación, una CPU y tarjetas de entradas y salidas tanto analógicas como digitales. Sin embargo, todo este hardware se podría reducir a un menor número de tarjetas de entradas y salidas, además de poder prescindir de la CPU (el elemento más costoso) del PLC utilizando un software que emule esta CPU en el ordenador utilizando los recursos del mismo.

Por otro lado, se podría dar una nueva funcionalidad al sistema de adquisición de datos, dotar al equipo de la posibilidad de almacenar los valores de presión y eventos producidos con un muestreo de 1 segundo y que fuesen almacenados en una tabla Excel para su posterior supervisión. Junto a este desarrollo se propone la posibilidad de desarrollar el sistema utilizando la herramienta LabView, que permite una mayor potencia en lo que se refiere a adquisición de datos y análisis matemático.

Glosario

AWL	Listas de instrucciones
DB	Bloque de Datos
FC	Función lógica sin memoria
FB	Función lógica con memoria
FUP	Funciones de bloques lógicos
HMI	Human-Machine Interface
IEC	International Electrotechnical Commission
KOP	Lenguaje de contactos
MMI	Man-Machine Interface
MPI	Multi Point Interface
MTU	Master Terminal Unit
OB	Bloque de organización
PID	Proporcional Integral Derivativo
PLC	Programmable Logic Controller
RTU	Remote Terminal Unit
SCADA	Supervisory Control And Data Acquisition

Bibliografía

[1] Balcells, J. y Romeral, J. (1997). Introducción al control industrial. En J, Balcells, *Autómatas Programables* (pp. 3-9). Barcelona: MARCOMBO.

[2] http://www.elprisma.com/apuntes/ingenieria_quimica/regulaciondeprocesos/default.asp, accedido en Diciembre 2010.

[3] Balcells, J. y Romeral, J. (1997). Aplicaciones de los PC industriales. En J, Balcells, *Autómatas Programables* (pp. 351-384). Barcelona: MARCOMBO.

[4] http://www.festo-didactic.com/es-es/, accedido en Noviembre 2010.

[5] http://support.automation.siemens.com, accedido en Enero 2011.

[6] http://www.disa.bi.ehu.es/spanish/asignaturas/17270/P1.pdf, accedido Enero 2011.

[7] http://193.146.57.132/depeca/repositorio/asignaturas/30387/Tema5.pdf, accedido Enero 2011.

[8] Manual de producto SIMENS Simatic. *S7-300, CPU 31xC y CPU 31x, Datos técnicos*. Numero de referencia: 6ES7398-8FA10-8DA0.

[9] Martínez Torres, José. (1999). *Programación avanzada en Step 7.* Valencia.

[10] Balcells, J. y Romeral, J. (1997). Diseño de automatismos con señales analógicas. En J, Balcells, *Autómatas Programables* (pp.41-64). Barcelona: MARCOMBO.

[11] Manual de producto SIEMENS Simatic. *Software estándar para S7-300 y 400. PID Control (Regulación PID)*. Numero de referencia: C79000-G7078-C516-01.

[12] Manual de product SIEMENS Simatic. *S7-300 Instruction List.* Numero de referencia: 6ES7398-8FA10-8BA0.

[13] Ogata, Katsushiki. (1998). *Ingeniería de control moderna.* Mexico:PRENTICE-HALL.